JUNE 2018 M.Sc. in Electrical and Electronics Engineering NUR SYARAFİNA ZAHARİ

UNIVERSITY OF GAZİANTEP

GRADUATE SCHOOL OF

NATURAL & APPLIED SCIENCES

DEVELOPMENT OF VIBRATING MESH NEBULIZER DEVICE

M. Sc. THESIS

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

BY

NUR SYARAFINA ZAHARI

JUNE 2018

Development of

Vibrating Mesh Nebulizer Device

M.Sc. Thesis

in

Electrical and Electronics Engineering

University of Gaziantep

Supervisor

Prof. Dr. Ergun ERÇELEBİ

by

Nur Syarafina ZAHARİ

June 2018

REPUBLIC OF TURKEY
UNIVERSITY OF GAZİANTEP
GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Name of the thesis : Development of Vibrating Mesh Nebulizer Device

Name of the student : Nur Syarafina ZAHARİ

Exam date : 18/06/2018

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. A. Necmeddin YAZICI

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Ergun ERÇELEBİ

Head of Department

This is to certify that we have read this thesis and that in our consensus opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Ergun ERÇELEBİ

Supervisor

Examining Committee Members	Signature
Prof. Dr. Ergun ERÇELEBİ	....
Prof. Dr. İlyas EKER	....
Associate Prof. Dr. Ahmet Mete VURAL	....

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Nur Syarafina ZAHARİ

ABSTRACT

DEVELOPMENT OF VIBRATING MESH NEBULIZER DEVICE

ZAHARİ, Nur Syarafina
M.Sc. in Electrical and Electronics Eng.
Supervisor: Prof. Dr. Ergun ERÇELEBİ
June 2018
84 pages

Nebulizing therapy has been an efficient method for the treatment of patients suffering from pulmonary diseases. In the nebulization treatment, medical solutions and suspensions are transformed into aerosol particles and delivered directly to the patients' airways by the nebulizer. There are three types of nebulizers; jet nebulizers, ultrasonic nebulizers, and mesh nebulizers. Jet nebulizers have disadvantages of bulk in size and requiring a power source to operate, whereas ultrasonic nebulizers generate large residual volume, unable to aerosolize viscous solutions, and cannot be used for heat-sensitive solutions. In this thesis, in order not to encounter these disadvantages, a vibrating mesh nebulizer device by using the microcontroller and micropore mesh plate has been proposed. The design of the device consists of an electronic circuit that generates an analog signal with the specific voltage, frequency, and waveform in order to drive the micropore mesh plate. When the signal is generated, the mesh plate vibrates at a very high rate of speed and the liquid medication is drawn from the holes perforated on the mesh plate. The vibrating mesh nebulizer developed in this study has advantages in terms of compact size, portable, silent operation, low frequency, low power consumption, generates less residual volume, and can also be used for heat-sensitive solutions.

Key Words: Vibrating mesh nebulizer, piezoelectric transducer, aerosol generator, nebulizing therapy, medical devices.

ÖZET

TİTREŞİMLİ MESH NEBULİZATÖR CİHAZININ GELİŞTİRİLMESİ

ZAHARİ, Nur Syarafina
Yüksek Lisans Tezi, Elektrik-Elektronik Müh. Bölümü
Tez Yöneticisi: Prof. Dr. Ergun ERÇELEBİ
Haziran 2018
84 sayfa

Nebülizasyon tedavisi, akciğer hastalığı olan hastaların tedavisinde etkili bir yöntem olmuştur. Nebulizasyon tedavisinde, tıbbi solüsyonlar ve süspansiyonlar aerosol partiküllerine dönüştürülür ve nebülizatör ile hastaların hava yollarına doğrudan iletilir. Nebulizatörler temel olarak üç tipe ayrılır; jet nebulizatörler, ultrasonik nebulizatörler, ve mesh nebulizatörler. Jet nebulizatörler hacimde ve çalışması için ihtiyaç duyuduğu güç kaynağının büyük olmasının dezavantajına sahiptir. Ayrıca, ultrasonik nebülizörler, kıvamlı solüsyonları aerosol haline getiremeyen ve ısıya duyarlı çözeltiler için kullanılamayan büyük bir artık hacim üretirler. Bu tezde, bu dezavantajlarla karşılaşmamak için mikrodenetleyici ve mikro gözenekli örgü levha kullanılarak titreşimli örgü nebulizatör tasarımı önerilmiştir. Tasarım, mikro gözenekli örgü levhasını sürmek için belirli voltaj, frekans ve dalga formuna sahip bir analog sinyal üreten bir elektronik devreden oluşur. Sinyal oluşturulduğunda, örgülü plaka çok yüksek bir hızda titreşir ve sıvı ilaç, örgü plakasındaki deliklerden çekilir. Bu çalışmada geliştirilen titreşimli örgü nebülizatörün kompakt boyut, portatif, sessiz çalışma, düşük frekans ve düşük güç tüketimi açısından avantajları vardır, daha az kalıntı hacim oluşturur ve ısıya duyarlı çözeltiler için de kullanılabilir.

Anahtar Kelimeler: Titreşimli mesh nebulizatör, piezoelektrik transdüser, aerosol üreteci, nebülizasyon tedavisi, tıbbi cihazlar.

To My Beloved Parents, Brothers, and Sisters

ACKNOWLEDGEMENTS

In the name of Allah, the Most Gracious, the Most Merciful, I would like to express my gratitude to Allah s.w.t for the strength, energy, patience, and time that He has given to me in order to complete my study.

I would like to convey my gratitude to my supervisor, Prof. Dr. Ergun ERÇELEBİ for his guidance, advice, ideas, time, support and encouragement. This project would never be completed successfully without his assistance and anticipation. I also would like to thank the lecturers, my friends and those who have directly or indirectly contributed in completing this project.

Last but not least, I would like to express my sincerest thanks to my family for their spiritual and financial support towards the completion of my study. Without their help and support, this thesis will not be as it is today. May Allah bless them all, Insha-Allah.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF SYMBOLS/ABREVIATIONS

CF	Cystic Fibrosis
COPD	Chronic Obstructive Pulmonary Disease
pMDI	Pressurized metered dose inhaler
DPI	Dry powder inhaler
PIC	Peripheral Interface Controller
MCC	MPLAB Code Configurator
ANFIS	Adaptive neuro-fuzzy inference system
ANN	Artificial neural networks
NIPPV	Non-invasive positive pressure ventilation
XLP	Extreme low power
GUI	Graphical User Interface
MCU	Microcontroller
PWM	Pulse Width Modulator
OPA	Operational Amplifier
CMP	Comparator

DAC	Digital-to-Analog Converter
ADC	Analog-to-Digital Converter
CLC	Configurable Logic Cell
TMR	Timer
NCO	Numerically Controlled Oscillator
DDS	Direct Digital Synthesis

CHAPTER 1

INTRODUCTION

Since last few decades, nebulizing therapy has been used as one of the major treatments for patients suffering from pulmonary diseases. Nebulizer, a medical device that is used in nebulizing therapy will transform the medical solutions and suspensions into micro-sized aerosol particles and will deliver them directly to the patients' airway surfaces for effective nebulization process. Nebulizers are commonly used for patients' suffering from Cystic Fibrosis (CF), Asthma, Chronic Obstructive Pulmonary Disease (COPD), and other severe forms of lung infections and diseases. Usually in a case where the patient cannot coordinate the required inhalation and actuation specification for using the pressurized metered-dose inhaler (pMDI), or the necessary inspiratory flow required by dry powder inhaler (DPI) cannot be provided by the patient, nebulizer has been used as a better choice during the nebulizing therapy.

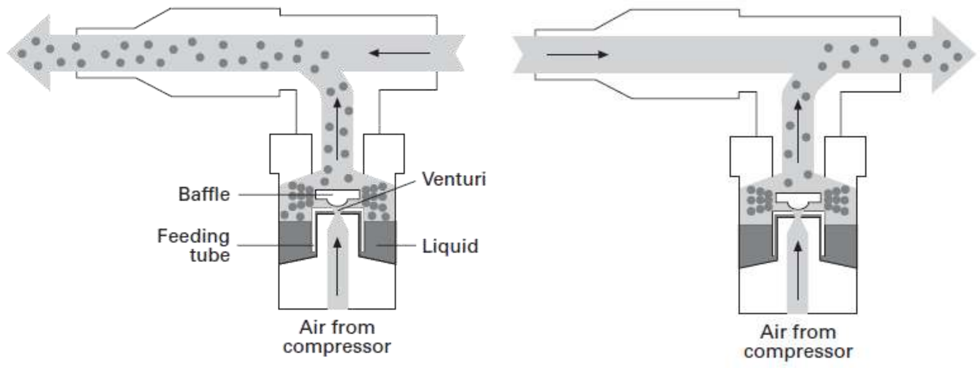
The word nebulizer, which originally comes from Latin word "*nebula*" that means mist, was first used in 1872 (Muers, 1997). In 1874, nebulizer was defined as "an instrument for converting a liquid into a fine spray, especially for medical purposes." In the earliest treatment of lung diseases, the herbs and medicine were delivered to the lung by way of smoking or steaming. It was depended on the mouth suction to inhale the medicine into the lung. Then in the 19th century, nebulizers with mechanical pumps for generating the gas flow for nebulization were invented, followed by nebulizers with electrical compressors were developed in the 1930s. These nebulizers generated aerosol particles with different sizes and most of the aerosol generated were non-respirable. This leads to the design of better nebulizer that can generate very fine particles by using Venturi system that was also employed in the conventional jet nebulizer. "The device that was invented consists of two glass tubes that have capillary openings at one end. The two ends being placed almost at a right angle with each other. The bigger opening end of the perpendicular tube is immersed in the medicated fluid. When the compressed air is forced through the horizontal tube, the air in the perpendicular one becomes exhausted, and the medicated solution then rises in it. When it arrives at the capillary opening, it is dispersed into very fine spray by the force of the compressed air along the other tube" (Anderson, 2005).

1.1 TYPES OF NEBULIZERS

Nebulizers generally can be divided into three types; jet nebulizers, ultrasonic nebulizers, and mesh nebulizers. These three types of nebulizers used different applications and ways in order to generate aerosols. Jet nebulizers used the application of compressed air whereas ultrasonic nebulizers used a piezoelectric crystal that vibrates at high frequencies. The latest type of nebulizers, which is mesh nebulizers used the micropump technology to produce aerosols.

1.1.1 JET NEBULIZERS

Jet nebulizers are a type of nebulizers that have been used mostly in the hospital and health center for the treatment of pulmonary diseases. These nebulizers used compressed air for generating the medical aerosol. In a conventional jet nebulizer, the driving gas from the compressor passes through Venturi (a very narrow hole) due to the high-pressure system as shown in Figure 1.1. A cone-shaped front was produced at the Venturi due to the falling pressure and greatly increased velocity. When the gas passes through the end of the narrow liquid feeding tube at a high velocity, a negative pressure was created. This resulted in the liquid is being sucked up by the Bernoulli effect and the liquid is drawn out into fine ligaments. Under the influence of surface tension, the ligaments are broken up into droplets. This usually generates droplets with the diameter of 15 - 50 µm. The larger droplets will impact on the baffles whereas the smaller droplets may be inhaled by the patient or may land on internal walls which eventually be returned to the reservoir for renebulization. The droplet size partly affected by the baffle design (O'callaghan and Barry, 1997).



(a) Inhalation (b) Exhalation

Figure 1.1 Conventional jet nebulizer design

Figure 1.1 shows the conventional jet nebulizer design and the conditions of the device mechanism during patient's inhalation and exhalation. During inhalation as shown in Figure 1.1(a), the negative pressure created due to the high-velocity gas passes through the Venturi sucks the fluid up and generate the aerosols. The small droplets generated are channeled out of the chamber and delivered to the patient's airways. During exhalation, the small droplets generated are released into the air as shown in Figure 1.1(b). Since the droplets are continuously generated, most of the aerosols are wasted during exhalation resulting in inefficiency of the device due to the low amount of aerosols that successfully deposited into the lung of the patient.

The conventional jet nebulizer also can be difficult to be used and not portable because of its need for compressed gas and additional tubing. Some studies also have proved that conventional jet nebulizer is inefficient in aerosol drug delivery as most of the primary droplets are caught on the internal baffles and structures (Nerbrink et al., 1994; Hess et al., 1996; Smith et al., 1995). Because of that, the design of jet nebulizers has been innovated over the years. The new jet nebulizers were designed by either utilizing baffles or having reservoirs that conserve the aerosols. This will reduce the size of large particles and enhance the efficiency of aerosol therapy. The latest design also has breath-enhanced aerosol production that results in greater drug delivery, a larger fine-particle fraction, and less drug loss during expiration (Ari, 2014).

The development of jet nebulizers design can be divided into four categories; jet nebulizers with a reservoir tube, jet nebulizers with a collection bag or elastomeric reservoir ball, breath-enhanced jet nebulizers, and breath-actuated jet nebulizers.

- Jet Nebulizers with a Reservoir Tube

  This type of nebulizers is the most widely used and least expensive nebulizer. This nebulizer provides a conventional constant-output nebulizer that generates aerosol continuously during inspiration, expiration, and breath-hold. Due to the release of aerosol during expiration and breath-hold condition, usually only 10-20% of the emitted aerosol is inhaled by the patient. This type of nebulizers has been considered as inefficient due to the low percentage of aerosol inhaled by the patient. This type of nebulizers is also not portable and requires compressed air or gas to operate.

- Jet Nebulizers with a Collection Bag or Elastomeric Reservoir Ball

  This type of nebulizers generates aerosol by continuously filling a reservoir. This nebulizer is considered as a dosimetric nebulizer which releases aerosol only during inspiration. During inspiration, the generated aerosols from the reservoir are inhaled by the patient through a one-way inspiratory valve that located between the mouthpiece and the collection bag. During expiration, the aerosols generated are stored in the collection bag and delivered to the patient in the next inspiration.

- Breath-enhanced Jet Nebulizers

  In the breath-enhanced jet nebulizers, aerosols are generated by the negative pressure from the patient's inspiratory effort. This type of nebulizers releases more aerosol during inhalation by using two one-way valves that will also prevent the loss of aerosol to the environment. During inhalation, the inspiratory valve opens and gas vents through the nebulizer whereas during exhalation, the gas passes through the expiratory valve in the mouthpiece.

- Breath-actuated Jet Nebulizers

  Breath-actuated jet nebulizers are designed to increase aerosol drug delivery by sensing the patient's inspiratory flow and generating the aerosol only during inspiration. This leads to the reduced loss of medication during the nebulizing therapy. However, these nebulizers require longer treatment time in order to achieve efficient nebulizing therapy.

Although there are many improvements made to improve the efficiency of jet nebulizers, these nebulizers are still large in terms of the size and require a power source to operate. Other than that, jet nebulizers also generate large residual volume because of the aerosolized droplets and solvent vapor that saturates the outgoing air. This leads to the cooling of drug solution and increasing solute concentration in the residual volume. Some of the disadvantages of jet nebulizer have been solved by using the ultrasonic nebulizer.

1.1.2 ULTRASONIC NEBULIZERS

Ultrasonic nebulizer is a type of nebulizer that uses a piezoelectric crystal that vibrates at high frequencies to produce aerosol particles. The frequencies used are usually more than 1MHz. In ultrasonic nebulizers, standing waves are formed when the ultrasonic vibrations from the piezoelectric crystal are transmitted to the surface of the drug solution. Then the crests of these waves are broken up into aerosol particles. The droplets size is inversely proportional to two-thirds of the power of the acoustic frequency. The aerosol particles generated are usually delivered to the patient by using a fan. Ultrasonic nebulizer is generally designed as shown in Figure 1.2.

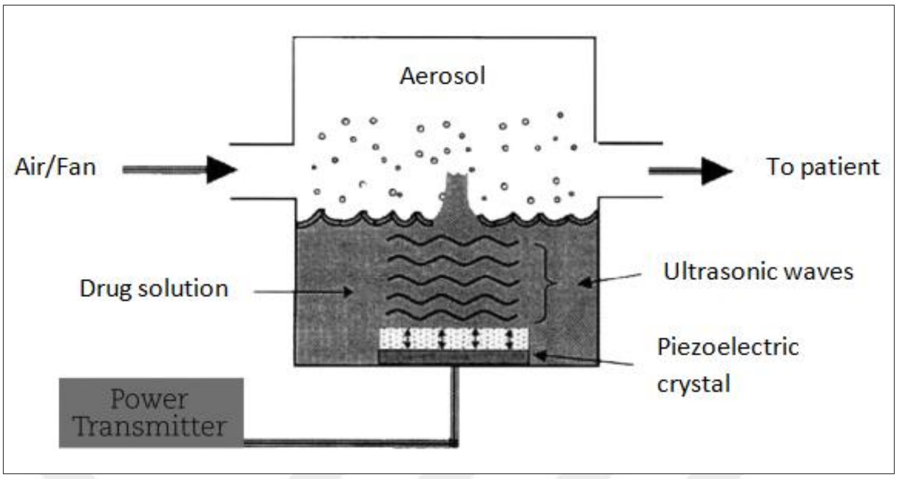


Figure 1.2 Ultrasonic nebulizer design

There is also ultrasonic nebulizer designed with baffles as shown in Figure 1.3. As in jet nebulizer, the use of baffles will remove the large droplets and the droplets will fall back into the medication cup. Ultrasonic nebulizers have more limitations compared to the jet nebulizers. The conventional ultrasonic nebulizers are expensive, bulky, and have a tendency to malfunction. Although newer models of ultrasonic nebulizers have been developed to encounter those problems, they still have limitations of large residual volume, unable to aerosolize viscous solutions, and degradation of heat-sensitive materials. This type of nebulizers also tends to increase the temperature of the medication solution. Because of that, ultrasonic nebulizers are not suitable to be used with suspensions and proteins (Yeo et al., 2010).

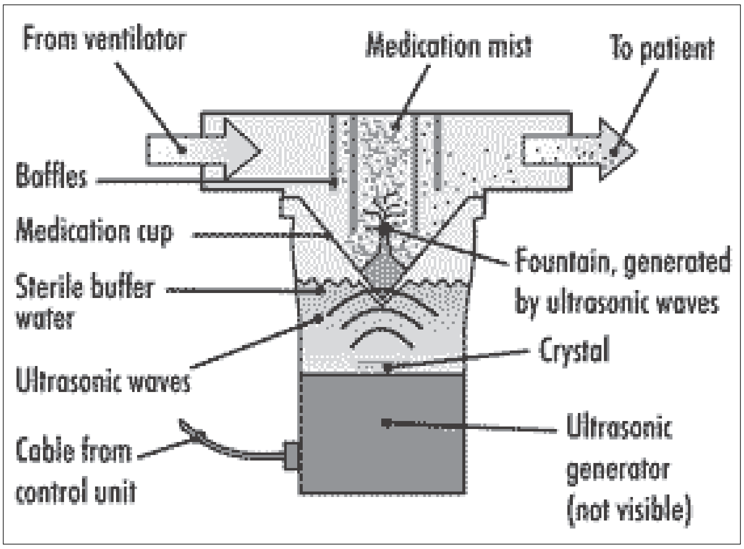


Figure 1.3 Ultrasonic nebulizer design with baffles

1.1.3 MESH NEBULIZER

Mesh nebulizer is the latest design of small-volume nebulizers. This type of nebulizer uses micropump technology in order to produce aerosol. Mesh nebulizer generates aerosol by forcing the liquid medication through the thousands of micro-sized holes drilled on the mesh plate. The aerosols generated through the mesh plate have similar size to the diameter of the mesh aperture. In conventional nebulizers, due to large residual volume, diluents are added to the drug solution to increase the volume of solution to be aerosolized. But, in mesh nebulizer, diluents are not needed since it can aerosolize drug solution as little as 0.2mL. Mesh nebulizer can be divided into two categories; passive mesh nebulizer, and active mesh nebulizer.

Passive mesh nebulizer uses a piezoelectric horn to create vibrations in the liquid drug. The vibrations generated force up the liquid drug and push it through a static mesh as shown in Figure 1.4. The static mesh plate is made from electroformed alloy and usually has over 1000 holes in a very small area. This type of nebulizer is categorized as passive mesh nebulizer due to the non-vibrating mesh plate. In this nebulizer, the aerosol is not recycled as in jet and ultrasonic nebulizer. The aerosols generated from the static mesh are directly delivered to the patient's airway surfaces.

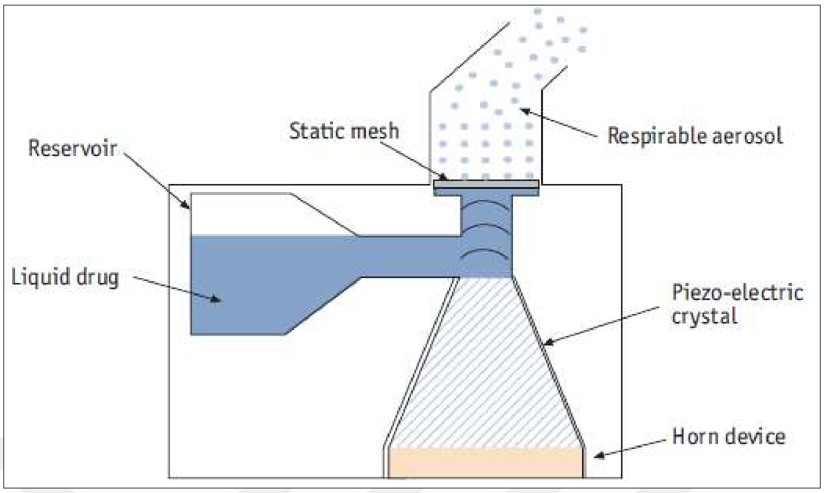


Figure 1.4 Passive mesh nebulizer design

Active mesh nebulizer, or generally known as vibrating mesh nebulizer is basically composed of a liquid reservoir attached with a micropore mesh plate, and an electronic circuit board for driving the mesh plate that operated using batteries. The mesh plate consists of a stainless steel plate that has been perforated with thousands of precision-formed, laser-drilled conical holes, and surrounded by piezoelectric ceramic. When the driver board generates an analog signal with the specific voltage, frequency, and waveform, the piezoelectric ceramic will vibrate at a very high rate of speed. As a result, the liquid medication is drawn through the holes to form fine particles of aerosol with uniform size that are delivered at a low velocity to be inhaled directly into the lungs of the patients. The design of vibrating mesh nebulizer is shown in Figure 1.5.

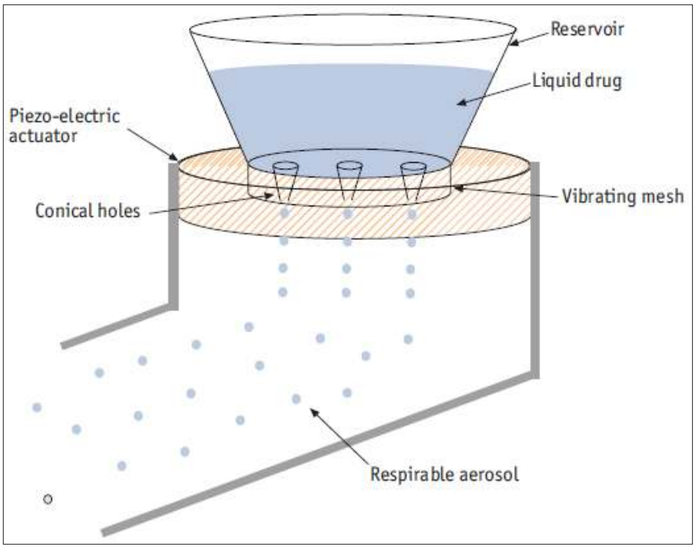


Figure 1.5 Active mesh nebulizer design

1.2 PROBLEM STATEMENT

Nebulizers have been commonly used for the treatments of patients suffering from pulmonary diseases such as Cystic Fibrosis (CF), Asthma, and Chronic Obstructive Pulmonary Disease (COPD). Nebulizers have been preferred over pressurized metered-dose inhaler (pMDI) and dry powder inhaler (DPI) due to its efficiency and ease of use, especially when the patient cannot coordinate the inhalation and actuation required for pMDI or the necessary inspiratory flow required for the DPI. There are three types of nebulizers; jet nebulizers, ultrasonic nebulizers, and mesh nebulizers. Jet nebulizers have limitations of bulky size and require a power source, whereas ultrasonic nebulizers have disadvantages of large residual volume and degradation of heat-sensitive materials.

There are a few studies have been done to improve the efficiency of jet nebulizer and ultrasonic nebulizer. These studies were discussed in Chapter 2. Vibrating mesh nebulizer is the latest design of nebulizer that can encounter the limitations of jet and ultrasonic nebulizer. However, up till today, there is very limited research regarding the design of this type of nebulizer. The commercialized mesh nebulizer also has varied performance in terms of efficiency. Hence, there should be more research done focusing on this type of nebulizer in order to develop more efficient nebulizer device.

1.3 THESIS OBJECTIVES

Due to the need for more research focusing on the mesh nebulizer, an attempt of studying and developing vibrating mesh nebulizer device has been done in this study. The main objective of this study is to develop a vibrating mesh nebulizer device that has better performance than jet nebulizer and ultrasonic nebulizer. In order to develop the vibrating mesh nebulizer device, the research done related to the design of all types of nebulizers have been studied and reviewed. In designing a vibrating mesh nebulizer, generating an adequate and reliable analog driving signal that is required by the piezoelectric ceramic on the micropore mesh plate has been one of the challenges related to the design. The mesh plate requires a high voltage waveform at a specific resonant frequency and the frequency may vary from device to device due to the manufacturing tolerances. Hence, in linear with the main objective, this study aims to develop an electronic circuit and system that drives the micropore mesh plate in order to generate aerosol. This study focuses mainly on the development of the electronic circuit and system part. So, the design of the micropore mesh plate is not discussed in details. However, the reviews of some mesh plate design that have been presented in other research are discussed in Chapter 2.

1.4 ORGANIZATION OF THESIS

This thesis composed of six chapters. In Chapter 1, introduction and general information regarding nebulizers such as definition, history, and type of nebulizers were discussed. Then the problem statement and objectives of the thesis were also presented in this chapter.

In the following Chapter 2, the reviews of the research that have been done in the past few years regarding jet, ultrasonic, and mesh nebulizers were presented. Other than the research regarding the enhanced design of jet and ultrasonic nebulizers, some research done related to the mesh plate design are also discussed in this chapter. Then the general information regarding PIC16F1713, MPLAB Code Configurator (MCC) and PICkit 3 are discussed.

Chapter 3 focuses on the hardware design of the electronic circuit for driving the micropore mesh plate. In the hardware part, the circuit is divided into two stages; BOOST CONVERTER and SIGNAL GENERATOR STAGES.

Chapter 4 discusses about the software designs. In the software part, the system is programmed into six state machines; INITIALIZE, SWEEP, RUN, STOP, INDICATE, SLEEP.

In the next Chapter 5, the experimental setup of the circuit and the result are presented. Then, the advantages of mesh nebulizer and the comparison between jet, ultrasonic, and mesh nebulizer are discussed.

Finally, in Chapter 6, the conclusion of this project and also the recommendations for future research in this area of study are presented.

CHAPTER 2

LITERATURE REVIEW

2.1 RELATED WORKS

Over the past few decades, a lot of research has been done in the area of inhalation drug delivery by using nebulizers. Throughout the years, there are some research done to improve the efficiency of nebulizers in delivering drug to the patient through the inhalation. In these studies, new and enhanced designs of jet and ultrasonic nebulizers were developed in order to get better nebulization therapy results. Other than that, there are also some studies that discuss the design of the mesh plate.

2.1.1 JET AND ULTRASONIC NEBULIZER

In a research by Yardimci (2011), a new design of jet nebulizer by controlling the compressor power and treatment time period was presented. During mechanical ventilation, the effectiveness of aerosol drug delivery is influenced by the patient, ventilator, and nebulizer factors. The nebulizer efficiency itself can be measured based on the residual drug volume, particle size, and method of aerosol generation. In this research, adaptive neuro-fuzzy inference system (ANFIS) has been adapted for the compressor fuzzy control unit. The ANFIS, which utilizes the application of fuzzy logic and Artificial Neural Networks (ANN), extracts the membership function parameters from a data set that describes the system behavior. Then the ANFIS learns the features in the dataset and adjusts the system parameters according to a given error criterion. Two input and one output parameters were used to control the compressor out. The input parameters are compressor speed and delivery pressure whereas the output parameter is aerosol mass distribution. For the treatment period timing function, the microcontroller was used for the timing function where the patient can choose the designated treatment time by using push button. The system was tested by measuring the pressure-flow characteristics of the compressor and the size distribution of aerosol mass. A rotameter and a pressure transducer were used to determine the pressure-flow characteristics, which resulted in 140kPa of pressure and 6.3L/min of flow rate. According to the results, it can be expected that the developed fuzzy model can be used as an appropriate control system for the nebulizer compressor. The compressor speed should be controlled since higher bias flow will decrease the aerosol drug delivery.

In another research by Tsai et al. (2013), a pocket-sized ultrasonic nebulizer was designed by using the silicon-based MHz multiple-Fourier horn ultrasonic nozzle that has a capability of producing high-throughput micrometer-sized monodisperse droplets at low drive power. The nozzle was designed to vibrate in a single longitudinal mode at the resonance frequency of the three Fourier horns. Fused silica tube is used to externally transport the liquid to be atomized to the nozzle end face. When the vibration displacement of the nozzle end face exceeds the onset threshold, droplets are ejected from the liquid surface. Due to the single vibration mode at the single resonance frequency that ensures the single-mode Faraday wave excitation and amplification, monodisperse droplets at very low electrical drive power were produced. The low drive power eliminates the temperature rise that may damage the heat-sensitive drug liquids. The nebulizer design consists of a nozzle, IC electronic driver, cell-phone battery, micro pump, drug reservoir, and liquid feed. The nebulizer generates monodisperse medicinal aerosols with a diameter ranging from 2.2 to 4.6μm, at throughput as high as 200μl/min and as low as 0.3W electrical drive power. The nebulizer also has been tested and compared with some commercial devices and the result showed that this nebulizer has a better performance. The device is also less prone to clogging since no mesh is used in generating the medicinal aerosols.

A pressure actuated synchronization system to inject nebulized drug only during the inspiration phase of breath was developed in a research by Staderini and Rusu (2014). A BPAP ventilator, a flow sensor, a vibrating mesh technology nebulizer, filters, a passive exhalation port, pipes and ties to link a mechanical test lung were used to set up a situation similar with the in vivo non-invasive positive pressure ventilation (NIPPV) therapeutic method. A pressure sensor was used to provide a signal for activation/deactivation of the nebulizer. When an inspiration begins, the pressure sensor detects the change in pressure of the flow and activates the nebulizer using the embedded electronic system. When another change of pressure occurs during expiration, the activity of the nebulizer is stopped. The embedded electronic system consists of a low power single supply operational amplifier, a voltage regulator, a timer, a relay, two light emitting diodes, and a potentiometer. The modified device operates in four different modes for each type of pressure (BiPAP or atmospheric). It is either can be operated only in the moment of inspiration, or it can be activated in the moment of expiration, or it can be stopped from nebulizing, or it can be nebulized continuously. With the presence of switches and different modes of operation, the system developed in this research could be suitable for different other applications.

Ivanova and Glazova (2015) had proposed an enhanced design of ultrasonic nebulizer targeting for children suffering from bronchial asthma in their research. The purpose of the research is to develop a digital registrator for ultrasonic nebulizer in order to monitor children's bronchial asthma attack frequency. Children suffering from bronchial asthma need a continuous monitoring of its development since asthma will progress rapidly if proper treatment is not given. The registrator would record the time parameters when the nebulizer was switched on, store the data and its output to electronic devices for further processing and transfer to the attending doctor. The block diagram of the interaction between a patient and a doctor by the registrator, a block diagram of the registrator and main unit of it, and a work algorithm were presented in the paper. The registrator contains the external data receiving device, clock-timer unit display, power supply unit, Bluetooth, data output device, and logger of events. The logger of events is the main registrator unit that receives information about the device enabling, records the time and date of event, stores it in memory for later transmission to output devices. In the logger of events, there are a DC-DC converter, clock-timer unit, microcontroller and internal memory unit. The development of the registrator can be a help to the doctor for providing essential treatment and dose of medication to the children suffering from bronchial asthma, based on the constant monitoring records from the registrator.

2.1.2 MICROPORE MESH PLATE DESIGN

In a research by Jeng et al. (2007), a piezoelectrically actuated nebulizer was fabricated by using micromachining techniques. The fabricated device consists of a piezoelectric actuator, an injection orifice-array plate, a front cover plate, and a drug reservoir. The piezoelectric actuator that located between the injection orifice plate and the reservoir transforms the electrical energy supplied by an external electrical pulse into mechanical energy. The piezoelectric actuator deflects and generates a pressure on the liquid drug in the reservoir when a voltage is applied to the piezoelectric actuator. The liquid in the reservoir is being squeezed through the array of orifices on the injection orifice plate due to the deflection of the piezoelectric actuator in the forward direction. Various experimental measurements and numerical simulations have been performed in this study in order to investigate the atomized droplet behavior and fluidic mechanics of the developed device. The results of this study showed that the diameter of aerosol particles can be reduced by increasing the operating frequency, but this will lead to the greater volume of air entering the reservoir that will degrade the ejection performance. The maximization of ejection flow rate can be done by setting the frequency of the operating voltage equal to the resonant frequency of the piezoelectric actuator. The amount of atomization also can be increased by increasing the driving voltage. These findings can be used as a reference in developing a more efficient mesh nebulizer design.

A design of medical micro-nebulizer has been proposed in a research by Shen et al. (2008). The design of the micro-nebulizer integrates a piezoelectric actuator, micro-nozzle plate, and the cavity of the micropump to get a high quality atomizing effect. The main objective of this research is to develop a power saving micro-nebulizer that can produce small particle size and a narrow range of particle size. In this research, the nozzle plate was fabricated by using the micro-electroplating technique. By using the micro-electroplating technique, the metals are electrodeposited into the structure defined by the photoresist mask. After the electroplating process, the photoresist masks were removed and the micro-electroplating was generated. The nozzle plate fabrication process started with the use of photoresist AZ4620 to define the column-like pattern, followed by electrodepositing of Ni as a substratum of the nozzle plate.

Then a backside etching by KOH was applied to remove the silicon pattern and AZ4520, and finished by electrodepositing of Au to complete the process. The 30µm thick nozzle plate contains 14000 holes distributed in an area of 1cm^2. The dimensions of the orifice are 80µm of fluid inlet diameter and 4 ± 0.5µm of fluid outlet diameter. The developed piezoelectric micro-nebulizer developed in this study was tested and compared with the commercial jet and ultrasonic nebulizer. Based on the results of the simulations and experiments conducted in this study, it showed that the droplet size distribution of the developed piezoelectric micro-nebulizer is finer and more uniform compared to the commercial jet and ultrasonic nebulizer used in this study.

In another research by Lin et al. (2011), an ultrasonic nebulizer using an electroformed palladium-nickel alloy nozzle plate, an RLC high-frequency driving circuit, and a piezoelectric transducer has been developed and tested. In this research, laser ablation technique and palladium-nickel alloy electroforming technique were used to fabricate the nozzle plate. The palladium-nickel was electro-deposited in both vertical and lateral growing directions. The thicker the electroformed palladium-nickel, the smaller the nozzle orifice size. This method can achieve a nozzle orifice dimension deviation less than 1µm. In this research, the electroformed nozzle plate was fabricated with a thickness of about 40µm and orifice size of 4µm. The hardness of the fabricated nozzle plate was also tested in order to prove that it is suitable for high frequent vibrating nebulizer system. The nozzle plate must be hard enough to endure the vibration and avoid the bending deformation of shape and fracture. The result showed that the hardness of the nozzle plate was close to 584Hv which proved that the developed electro-deposited palladium-nickel alloy is suitable for use in the nozzle plate of high frequent vibrating nebulizer system.

2.2 PIC16F1713 MICROCONTROLLER

PIC16F1713 is a microcontroller by Microchip Technology Incorporated that combine the Intelligent Analog integration with low cost and extreme low power (XLP) in order to serve a variety of general purpose applications. This 28-pin microcontroller delivers on-chip Op Amps, Core Independent Peripherals (CLC, NCO, and COG), Peripheral Pin Select and Zero Cross Detect that can be used for flexible design applications. This microcontroller also has 25 I/O pins and can operate at a voltage between 2.3V and 5.5V. The operating speed of this microcontroller is between 0-32MHz clock input with 125ns minimum instruction cycle.

Figure 2.1 PIC16F1713 chip

Table 2.1 Specifications of PIC16F1713

Product Attribute	Attribute Value
Maximum Clock Frequency	32 MHz
Flash Program Memory	4096 words
Data SRAM	512 bytes
High-Endurance Flash Data Memory	128 bytes
Number of I/Os	25
Number of 10-bit ADC channels	17
Number of 5/8-bit DAC	1/1
Number of High-Speed/Comparators	2

Number of Op Amp	2
Number of Timers	5
Number of PWM	2
Number of CLC	4
Number of NCO	1
Operating Supply Voltage	2.3V to 5.5V
Min / Max Operating Temperature	-40 °C / +85 °C
Debugging Methods	- Integrated on Chip - using Emulation Header

2.3 MPLAB CODE CONFIGURATOR

MPLAB Code Configurator (MCC) is a plug-in for MPLAB X IDE that generates driver code using a Graphical User Interface (GUI). The generated drivers control the peripherals on PIC microcontrollers. MCC simplifies the process of configuration setup of all the peripherals. The MCC allows the user to set up the peripherals in a GUI environment that provides an easier means for the user. The MCC also can be used for configuring and generating libraries, which allows the user to configure and generate code for software libraries and off-chip peripherals. The generated drivers or libraries can be used in any Microchip PIC device application program. By using MCC, the peripherals setup is done by selecting the options from the menus within the MCC, and then the MCC will generate source and header files based on the selections made in the GUI. The generated files are added to the active project of MPLAB X IDE.

2.3.1 MCC OPERATING AREAS

The MCC user interface consists of three main operating areas as shown in Figure 2.2.

1. **Resource Area**: This area displays the MCC supported on-chip peripherals, libraries, and off-chip peripherals that are available for the selected device. The Resource Area is divided into two sub-area; Project Resources and Device Resources. Project Resources area shows the resources selected for the project, whereas Device Resources area shows the available resources and the available MCC library versions on the local machine.

2. **Composer Area**: This area is the main area in which a peripheral or library driver can be configured. This area provides the user with an interface to configure the peripheral or the library driver.

3. **Pin Manager Area**: I/O pins of the device can be configured in this area. The Pin Manager Table View provides an interface for configuring the device pins. This area also displays the pins in a Package View, showing the pin out of the device and the functions assigned to the pins.

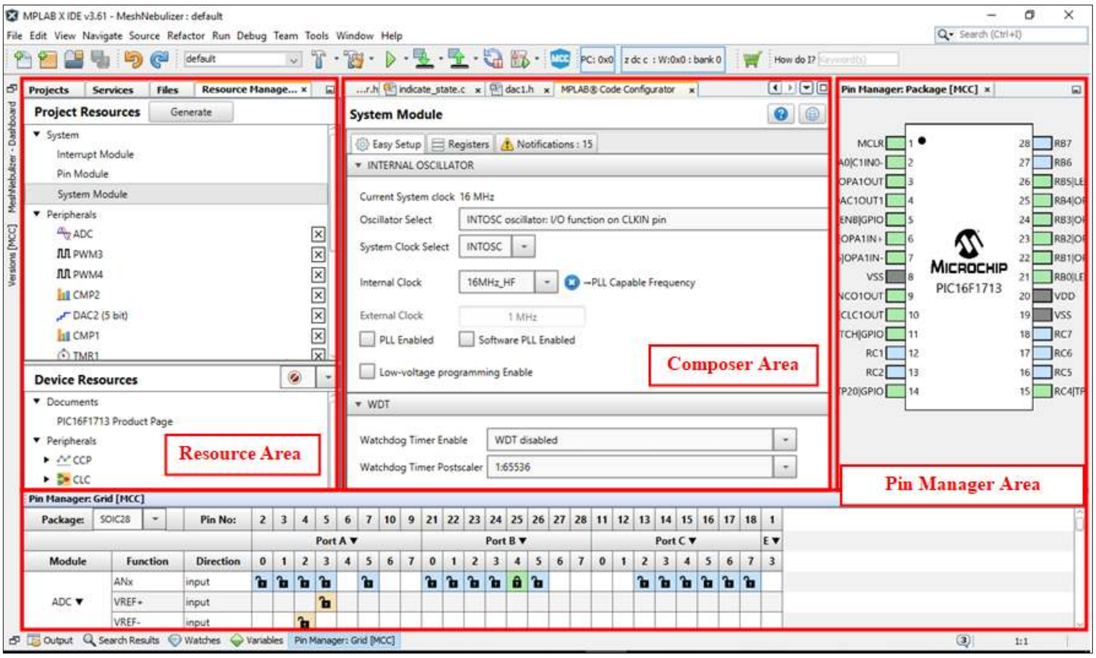


Figure 2.2 MCC operating areas

2.3.2 USING MCC TO SELECT AND SETUP PERIPHERALS

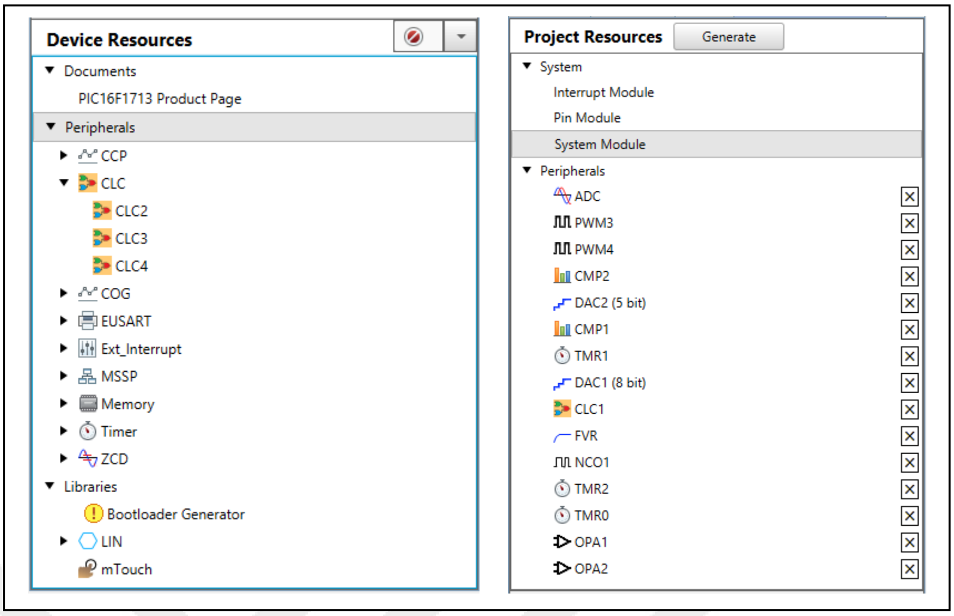


Figure 2.3 MCC Resource Area

By using MCC, it is easier to select the peripherals for a device. The Device Resources area shows the resources available on the device. By clicking on the name of the peripheral, a list of associated options including multiple versions of the peripheral will appear. For an example, in Figure 2.3 above, there are multiple CLCs on PIC16F1713. So, the submenu shows the list of all of CLCs that are available. After clicking on the specific peripheral, the peripheral will be added to the Project Resources area.

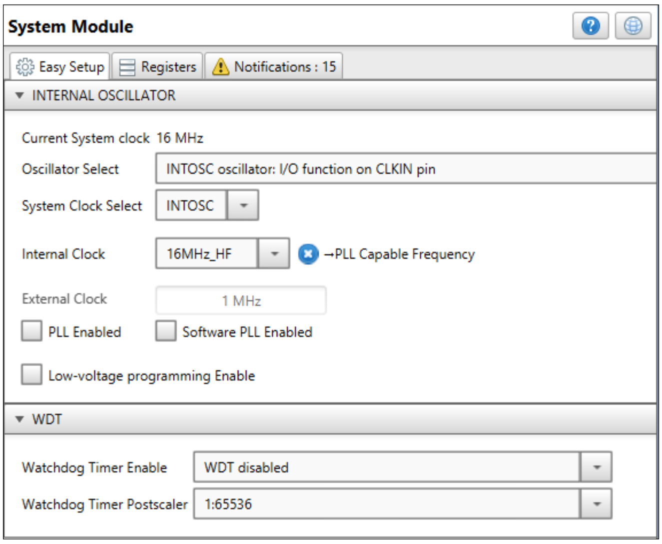


Figure 2.4 MCC Composer Area

The peripherals are set up in the Composer Area. When a peripheral is selected in the Project Resources area, the corresponding configuration GUI is displayed in the Composer Area. There are three tabs in Composer Area; Easy Setup, Registers, and Notifications.

1. Easy Setup: This is where the settings for the peripherals are selected through a variety of drop-down selections and checkboxes menu.

2. Registers: This tab shows the various registers for the particular peripheral. The configuration settings can also be viewed and modified under this tab.

3. Notifications: This tab shows the conflicts or recommendations associated with the peripherals setup. This may be as simple as an I/O pin is set to analog and is recommended to be set to digital for the peripheral setting selected.

2.3.3 USING MCC TO SELECT I/O PIN CONFIGURATION

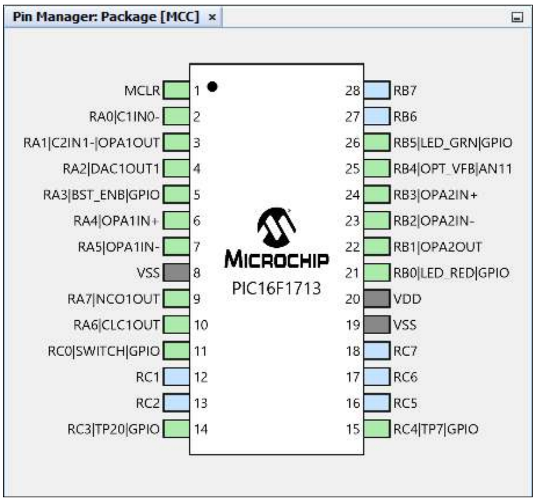


Figure 2.5 Pin Manager package view

I/O pin configurations can be set up using the GUI in Pin Manager area. First of all, the device package type to be used must be selected. Then a representation of the graphical picture of the device will appear in the Pin Manager package view area. The Pin Manager table view area can be used for selecting the arrangement of the I/O configurations.

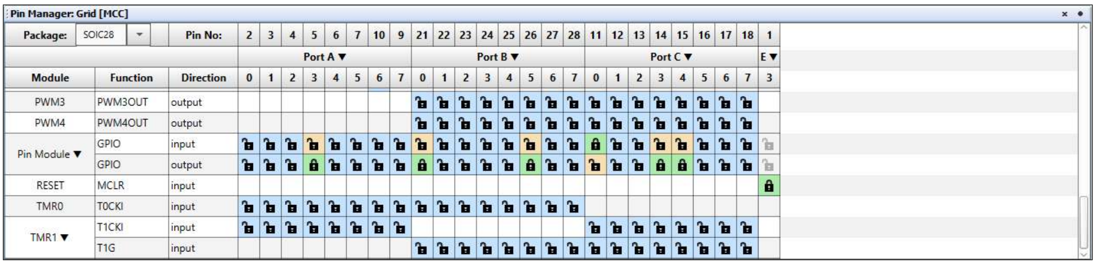


Figure 2.6 Pin Manager table view

In Figure 2.6 above, the menu of peripherals to connect the pin to can be positioned on the top (horizontal) or on the left side (vertical) by clicking on the horizontal or vertical selection. The pin(s) associated with the peripheral will show up in the row or column of the peripheral name. Hence it is easier to determine which pins to select for that particular peripheral. When a pin is selected, the color will change to green and the lock symbol will close. This indicates that the pin has been locked into that arrangement. Clicking on the green lock will reset it back to a blue, unused state. The explanations of the pin colors as shown in Figure 2.6 are as follows:

1. Gray colored pin: This indicates that the pin is not usable in the selected configuration and that there is no enabled module which has any functionality on that pin. There are also grayed out locks on a white background that indicate pins that are locked out by selected system functions.

2. Blue colored pin: This indicates pins that are available to be allocated to a module.

3. Green colored pin (with a lock): This combination indicates that the pin has been allocated and selected for a module. The name that appears on the pin is either the name of the pin in the module's context or a custom name entered.

4. Green colored pin (with chain link): This combination indicates that the pin is shared between multiple functions.

5. Yellow colored pin: This indicates that a possible alternate pin for an already allocated pin functions.

6. Grayed out locks on a white background: This indicates pins that are locked out by selected system functions.

2.3.4 GENERATING CODE

After each peripheral is set up along with the configuration and I/O, then just click on the Generate button in the Resource Area to have the MCC create all the peripheral driver code for the project. The code will be placed into the project automatically. If it is a new project, then a main.c file will also be produced. If the project already has a main.c file, then the MCC will on produce the peripheral driver files. The code will show up in the project area of MPLAB X IDE. The code produced is fully open sourced and editable. It can be used as a base and modified for the project needs.

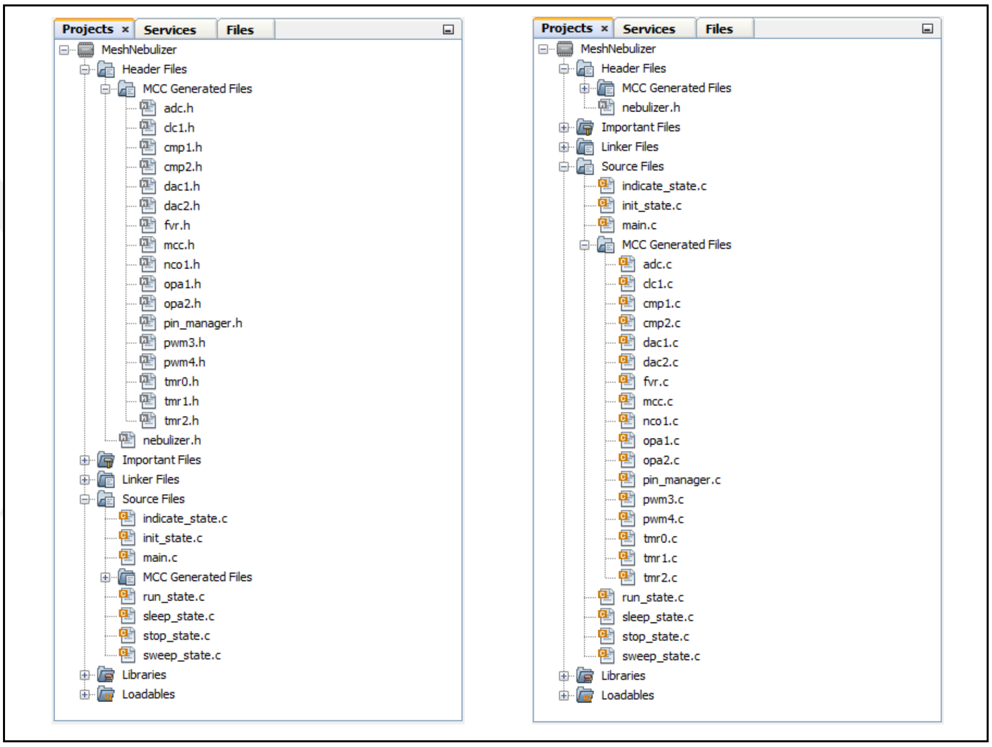


Figure 2.7 MCC generated header and source files

2.4 PICkit™ 3

PICkit 3 In-Circuit Debugger is the Microchip's low-cost hardware debugger/programmer. It can program and debug Microchip PIC microcontrollers (MCUs) and dsPIC digital signal controllers just as the MPLAB REAL ICE™ In-Circuit Emulator and MPLAB ICD 3. It also works seamlessly with MPLAB X IDE. PICkit 3 In-Circuit Debugger can be used as a development programmer, but not as a production programmer. The PICkit 3 is connected to the PC using a full-speed USB 2.0 interface and connected to the target using the In-Circuit Serial Programming interface. In the debugger system, the code is executed like an actual device due to the use of a device with built-in emulation circuitry. All available features of the target device can be accessed and modified through the MPLAB IDE interface. The PICkit 3 was developed for emulating embedded processors with debug facilities. The important features of PICkit 3 are as follows:

- Full-speed USB support using Windows standard drivers
- Real-time execution
- Processors run at maximum speeds
- Built-in over-voltage/short circuit monitor
- 1.8 - 5V range
- Diagnostic LEDs
- Read/write program and data memory of microcontroller
- Erase of all memory types with verification
- Peripheral freeze at breakpoint

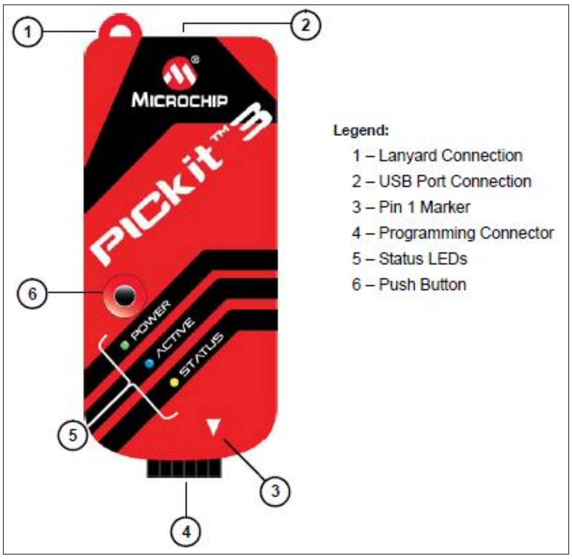


Figure 2.8 PICkit 3 In-Circuit Debugger

Figure 2.8 above shows the PICkit 3 In-Circuit Debugger.

1. **Lanyard Connection**: A convenient lanyard connection is available on the programmer.

2. **USB Port Connection**: PICkit 3 is connected to the PC through this USB port. The USB port connection type is USB mini-B type.

3. **Pin 1 Marker**: This marker is an indicator for the location of pin 1 of the programming connector.

4. **Programming Connector**: This programming connector is a 6-pin header that connects the PICkit 3 to the target device for programming purposes. Figure 2.9 shows the pinout specifications for connecting this programming connector to the PIC microcontroller pins.

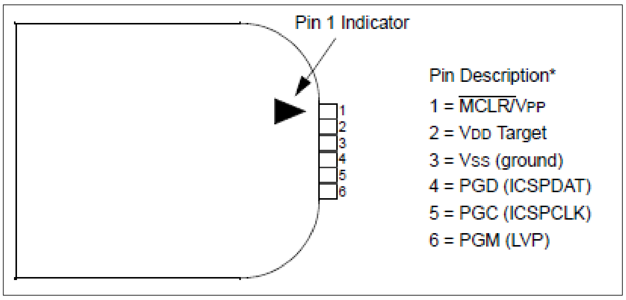


Figure 2.9 Programming connector pinout specifications

5. **Status LEDs**: The status LEDs indicates the status of PICkit 3.

    1) **Power** (green): Power is supplied to the PICkit 3 through the USB port.

    2) **Active** (blue): The PICkit 3 has been connected to the PC USB port and the communication link is active.

    3) **Status**:

        Busy (yellow): The PICkit 3 is busy with a function in progress, such as programming.

        Error (red): The PICkit 3 has encountered an error.

6. **Push Button**: The push-button is used for Programmer-To-Go.

CHAPTER 3

HARDWARE DESIGN

Figure 3.1 shows the block diagram of vibrating mesh nebulizer device developed in this study. The device design is divided into software and hardware part. The hardware part will be explained in this chapter and the software part will be explained in the next chapter. In the hardware part, PIC16F1713 has been used as the microcontroller in the circuit. This microcontroller is chosen due to its low cost, low power, and rich peripherals features. The hardware part is divided into two stages; Boost Converter Stage and Signal Generator Stage. In the Boost Converter Stage, the input voltage from the batteries is boosted to a high output voltage whereas, in the Signal Generator Stage, high volt sine wave signal at the certain resonant frequency is generated in order to drive the mesh plate for producing aerosol.

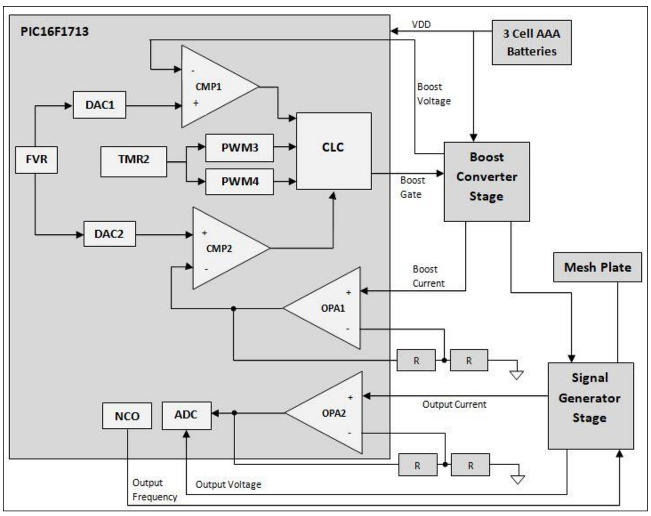


Figure 3.1 Block diagram of vibrating mesh nebulizer device

3.1 BOOST CONVERTER STAGE

Boost converter circuit is a circuit that converts the unregulated dc input to a controlled dc output at the desired voltage level. The basic boost converter circuit is shown in Figure 3.2(a). During one switching cycle, the switch (S1) transitions between a closed and opened position. If the switching cycle begins with S1 in the closed position as in Figure 3.2(b), the diode (D1) is reverse-biased. The circuit is divided into two loops; one with the battery (Vin) and the inductor (L1), and the other one with the capacitor (C1) and resistor (R1). The voltage across L1 is equal to the voltage supplied to the circuit, Vin. In this phase, the inductor L1 is being charged by the current flowing through the loop. The current ramps up and energy is stored in L1. The inductor voltage is also high since there is no transfer of voltage to the load during this time. The load requirements during this time are supplied by C1.

When S1 switches to open position as in Figure 3.2(c), D1 is forward-biased. The voltage across L1 is Vin minus Vout and the energy stored in L1 (as well as the input source) is transferred to the load and C1. The inductor current is reducing linearly and the capacitor C1 is charged during this phase. The higher output voltage produced in this phase is due to the supplied energy by both the input source and the inductor L1 to the load. To be summarized, during closed switch time, the inductor is charged and the capacitor delivers the required power to the load, whereas during opened switch time, the inductor is discharged by supplying the full power to the load and charging the capacitor simultaneously.

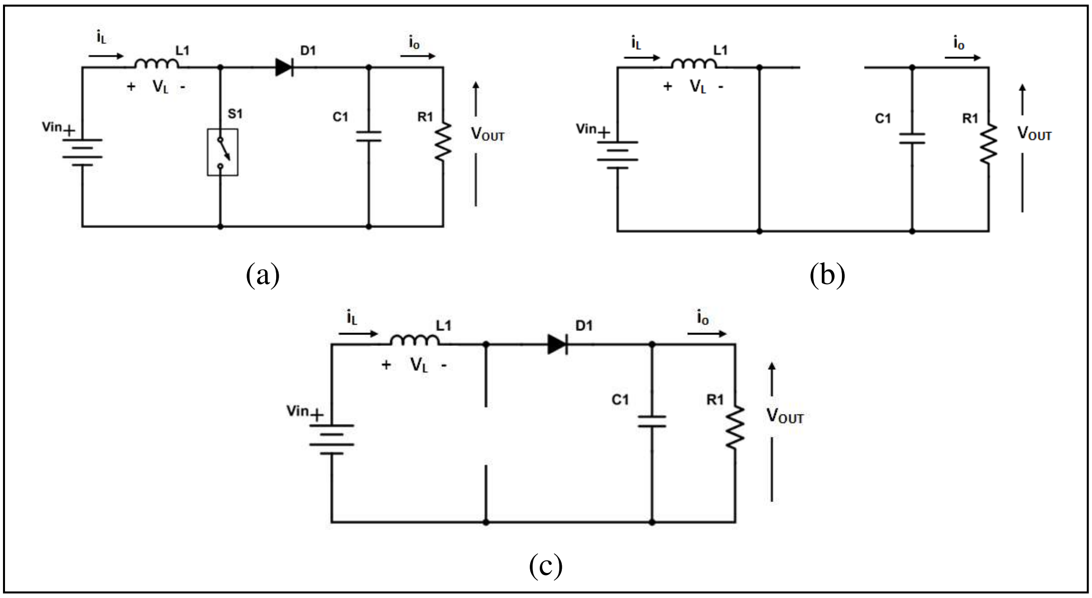


Figure 3.2 Basic boost topology

In a more practical application, the switch is replaced with an N-channel MOSFET, which is also applied in the circuit designed in this study. The MOSFET acts as a switch that transits between ON and OFF state. The gate of the MOSFET is controlled by the PIC16F1713 MCU. As shown in Figure 3.3, in this stage, there are two MOSFET used for controlling the circuit. The P-channel MOSFET (Q1) is used for controlling the power supplied to both BOOST CONVERTER and SIGNAL GENERATOR stages. Since in a boost converter circuit, the voltage supplied by the battery is always delivered to the load continuously, there is a need of switch of interrupting the voltage when it is not needed by the load. The voltage is supplied to the circuit by enabling the output pin on the PIC16F1713 MCU connected to the gate of MOSFET Q1. When the BST_ENB pin on the PIC16F1713 MCU is brought low, the voltage will be supplied to the circuit.

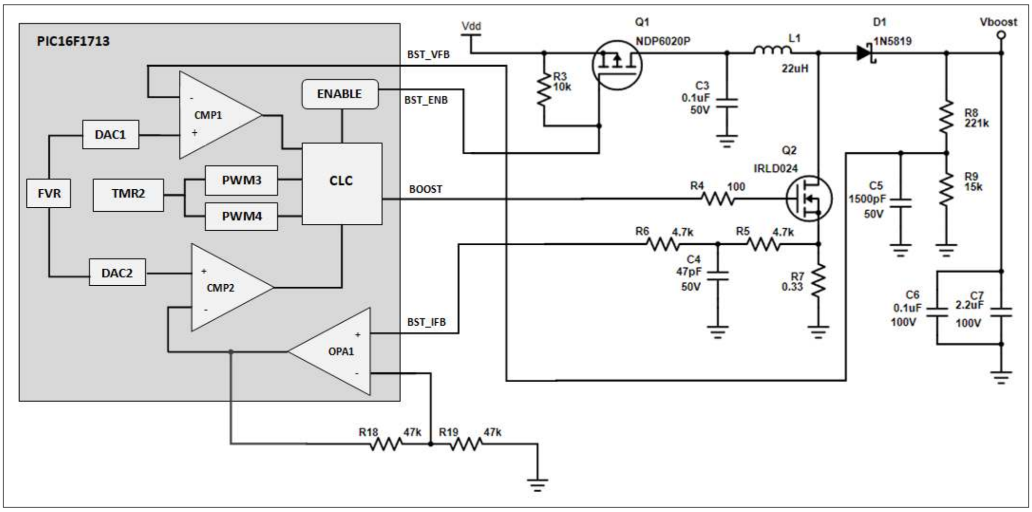


Figure 3.3 Boost Converter stage circuit

In this stage, two Pulse Width Modulator (PWM), an operational amplifier (OPA), two comparators (CMP), two Digital-to-Analog Converter (DAC), and a Configurable Logic Cell (CLC) modules in the PIC16F1713 MCU are implemented in the circuit to generate a boost regulator that runs completely independent from the software. The switching frequency is fixed by the timer (TMR2) and the two PWM modules at 100kHz. The PWM3 is set for a short duty cycle and the PWM4 is set for a long duty cycle. The boost duty cycle will start at the termination of PWM3 pulse and end at the termination of PWM4 pulse. The two PWM modules are used to delay the clock pulse to the flip-flop in the CLC modules in order to prevent the clock pulse from happening at the same time as the clear pulse.

A voltage divider formed by two different resistors (R8 and R9) is used to sense the output voltage (Vboost), and the voltage is fed back into the negative input of CMP1. The CMP1 will detect if the Vboost is high or low. The positive input of CMP1 is connected to the 8-bit DAC (DAC1), and DAC1 sets the output voltage of the boost converter stage. When the feedback voltage lower than the threshold value, boost pulses will be generated. The CLC output will drive the N-channel MOSFET Q2 and deliver additional current to the output. When the output rises above the threshold value, no pulses will be generated and the output voltage will decay as current is drawn from the capacitors (C6 and C7). Current sense resistor R7 is used to convert the inductor current to a voltage. The voltage from this resistor is amplified by OPA1 and is feed into the negative input of CMP2. The positive input of CMP2 is connected to the 5-bit DAC (DAC2). The output of CMP2 will terminate the boost cycle early if the peak current target is reached.

3.2 SIGNAL GENERATOR STAGE

In this stage, high-voltage sine wave signal is generated to drive the piezoelectric ceramic on the mesh plate. The circuit is basically an LC series circuit, with some special features. The voltage from the boost converter stage is also boosted in this stage in order to generate a higher output voltage. The circuit is constructed by connecting N-channel MOSFET Q3 to the Numerically Controlled Oscillator (NCO) module in the PIC16F1713 MCU. The NCO is used to drive the gate of MOSFET Q3. The NCO operates on the principle of Direct Digital Synthesis (DDS). DDS is a technique to generate an analog waveform from a time-varying signal in its digital form and a DAC. The NCO can be operated in two modes: fixed duty cycle PWM and frequency controlled Pulse mode. In this circuit, it is operated in fixed duty cycle PWM mode. A variable frequency with a fixed 50% duty cycle is provided by the NCO. This fine adjustment range of NCO allows the signal generator stage to be tuned to the particular mesh plate connected to the circuit.

Figure 3.4 shows the circuit designed for this signal generator stage. As shown in Figure 3.4, the output of NCO controls the gate of MOSFET Q3, and the Q3 controls the output generated for the piezoelectric ceramic on the mesh plate. The Q3 will pull the piezoelectric ceramic low, and when Q3 is turned off, the inductor L2 becomes a current source that drives the piezoelectric ceramic high. When Q3 is turned on, inductor L3 converts the sharp falling edge into sine wave as it forms a resonant circuit with the capacitor C11 and the piezoelectric ceramic. The other capacitor C10 acts as a DC blocking capacitor that keeps the voltage on the piezoelectric ceramic symmetrical and prevents the DC voltage on Vboost from affecting the piezoelectric ceramic when the output should not be driven.

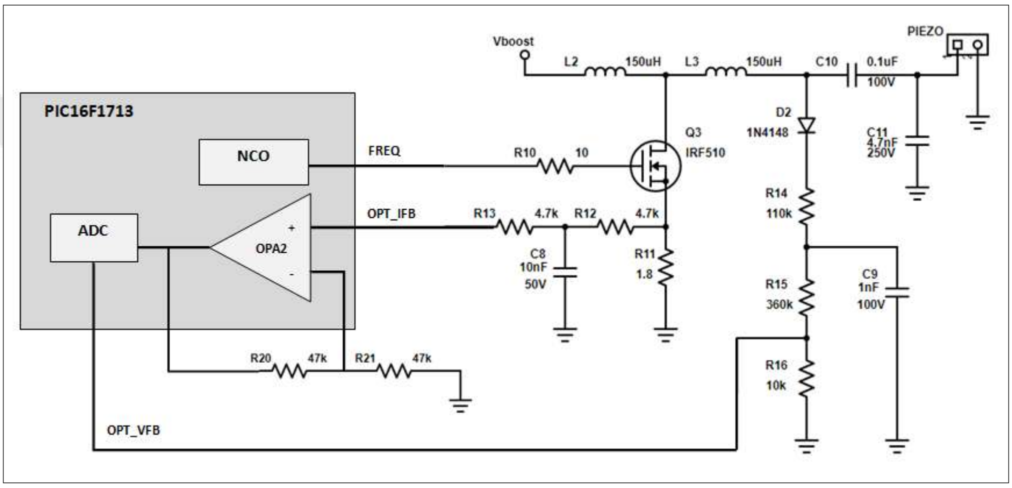


Figure 3.4 Signal Generator stage circuit

Current sense resistor R11 is used to measure the current in this stage. The current will rise when the circuit approaches resonance. This signal is used to detect the optimum output frequency during the frequency sweep. The voltage divider formed by resistors R15 and R16 is used to measure and monitor the output voltage generated for the piezoelectric ceramic.

3.3 MICROPORE MESH PLATE

The mesh plate used in this circuit is perforated with about 4000 of micro-sized, bell-shaped holes. In vibrating mesh nebulizer, the size of the holes has a major effect on the size of aerosol particles produced by the device. The smaller the holes, the finer the aerosol produced by the device. Figure 3.5 shows the mesh plate used in this circuit. The mesh plate is designed using a metal plate as the base as shown in Figure 3.5(a). The center of the plate (micropore area) is perforated with thousands of precision-formed, laser-drilled, micro-sized holes, and surrounded by the piezoelectric ceramic. Figure 3.5(b) is a picture of the mesh plate used in this circuit. The mesh plate operates at the nominal frequency of 113kHz and is manufactured with the features of low impedance, low dielectric loss, and resistant to corrosion.

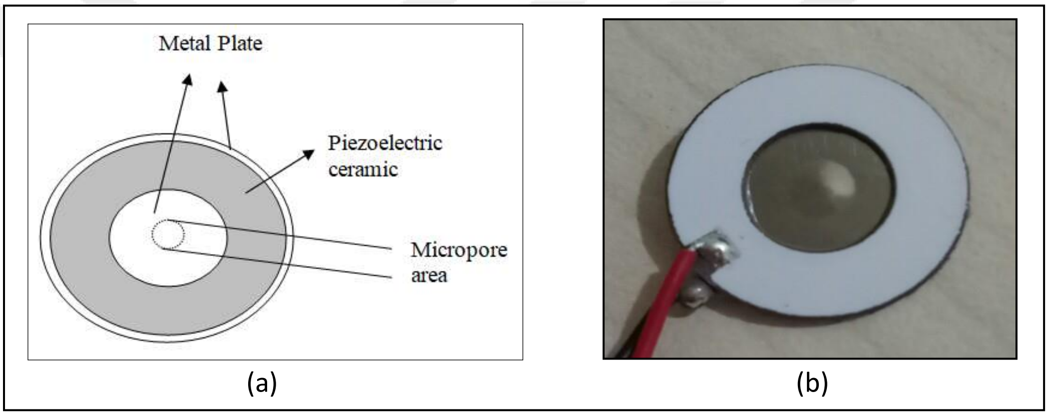


Figure 3.5 Micropore mesh plate

CHAPTER 4

SOFTWARE DESIGN

MPLAB X Integrated Development Environment (IDE) software is used for the programming part in the software implementation. MPLAB X IDE is a software program that runs on a PC (Windows, Mac OS, Linux) in order to develop applications for Microchip PIC microcontrollers and dsPIC digital signal controllers. For the setup of the configuration and of all the peripherals used in the circuit, MPLAB® Code Configurator (MCC) has been used.

4.1 MCC PERIPHERAL SETUP

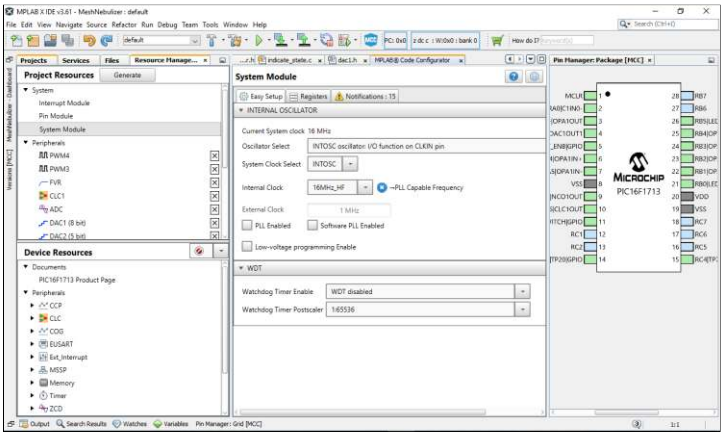


Figure 4.1 MCC peripheral setup

Figure 4.1 above shows the GUI for MCC Peripheral Setup. The first module in MCC that needs to be set up is System Module. After setting up this module, the setup of all the peripherals used has been proceeded, followed by the setup of Pin

Module. Since there is no interrupt used in this project, there is no need to setup the Interrupt Module.

4.1.1 System Module

In this module, the clock and several other peripherals that depend on the clock setting are set up. Internal oscillator is selected for the clock and it is selected to be run at 16MHz. This value is selected in order to get the best resolution for the PWMs and NCO. The configuration bits are also set up in this module.

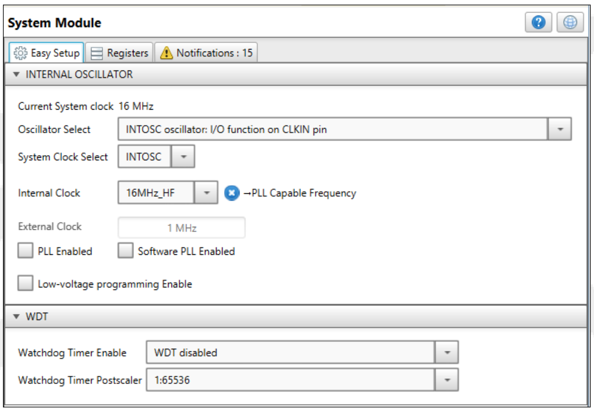


Figure 4.2 System Module setting

4.1.2 Timer

There are three timers used in this project. Timer0 is used for timing the LEDs and maximum treatment timer. Timer0 is an 8-bit timer/counter register. The timer is set up with 1:256 prescaler and the clock source is FOSC/4. The timer period is calculated as follows:

$$F_{TMR0} = \frac{F_{OSC}}{4 \times Prescaler \times 2^8} = \frac{16MHz}{4 \times 256 \times 256} = 61.035Hz \tag{4.1}$$

$$T_{TMR0} = \frac{1}{F_{TMR0}} = \frac{1}{61.035Hz} = 16.384ms \tag{4.2}$$

Where

F_{TMR0} is Timer0 frequency, F_{OSC} is oscillator frequency, and T_{TMR0} is Timer0 period.

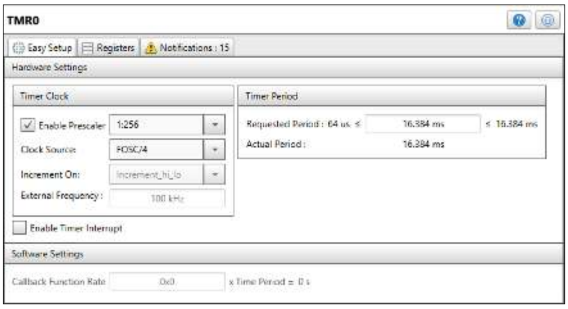


Figure 4.3 Timer0 setting

Timer1 is used for the delay in the project. Timer1 is a 16-bit timer/counter register. The timer is set up with 1:8 prescaler and clock source of FOSC/4.

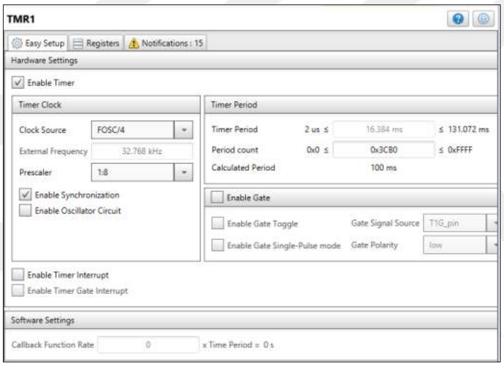


Figure 4.4 Timer1 setting

Timer2 is an 8-bit Timer and Period register. Timer2 is used as the period timer for the PWM modules. Timer2 is set up to overflow as close as possible to 100kHz since Boost Converter Stage is run at about 100kHz. 100kHz is equal to a period of 10µs.

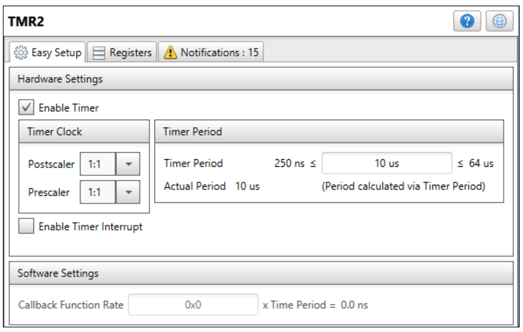


Figure 4.5 Timer2 setting

4.1.3 Pulse Width Modulator (PWM)

The two PWM modules, PWM3 and PWM4 are used in this project. In PWM3 setup, Timer2 is chosen as the timer associated with this PWM. A short duty cycle is assigned to PWM3 which is 10%. PWM3 is essentially a forced dead time in the boost output and the termination of this pulse starts the boost cycle.

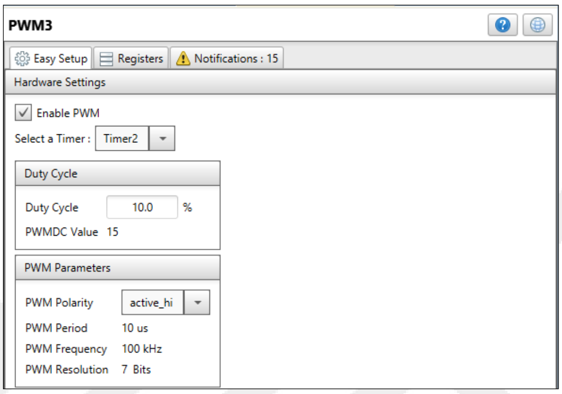


Figure 4.6 PWM3 setting

PWM4 sets the maximum duty cycle for the Boost Converter Stage. Hence, the duty cycle of this PWM4 should be set to as high as possible, but should be lower than 100%. Timer2 is also selected as the timer associated with this PWM4.

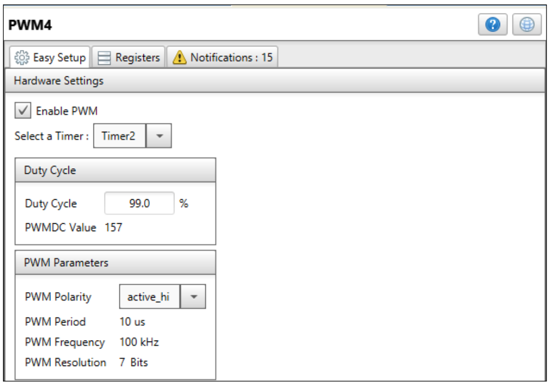


Figure 4.7 PWM4 setting

4.1.4 Fixed Voltage Reference (FVR)

The FVR is used as a reference for the DAC and ADC modules. Since the MCU is powered directly from the batteries, VDD will vary as the batteries age. Hence, VDD cannot be used as a reference for the DAC and ADC. FVR is a stable voltage reference that is independent of VDD. The output of FVR can be chosen by selecting the gain factor of the amplifier, which is either 1x, 2x, or 4x. This will result in 1.024V, 2.048V, or 4.096V of possible voltage level. Since the nebulizer will run at the lowest input voltage of 2.9V, so 2.048V is chosen as the reference value for DAC and ADC. This should be stable across the entire VDD range from the batteries.

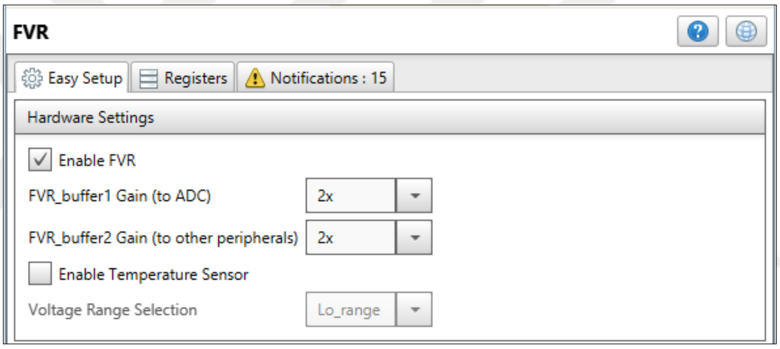


Figure 4.8 FVR setting

4.1.5 Digital-to-Analog Converter (DAC)

DAC is a peripheral for taking digital data and producing a variable output level. The PIC microcontroller output supplies a variable voltage output that is ratiometric with the input source. DAC1 which has 8-bit resolution is used to set the output voltage limit for the Boost Converter Stage. The positive reference of DAC1 is set to FVR and the negative reference is set to VSS.

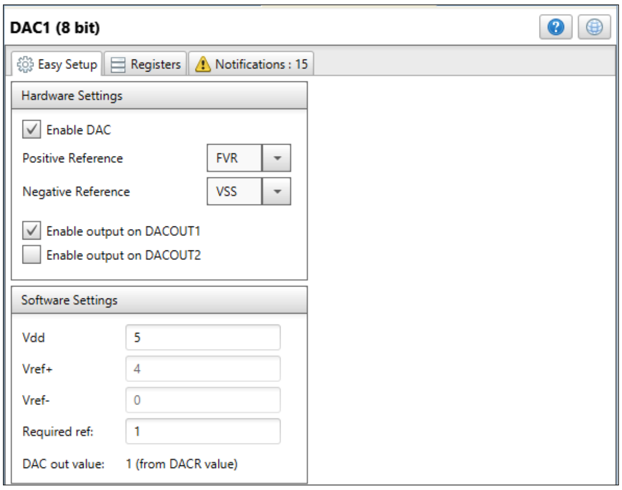


Figure 4.9 DAC1 setting

DAC2 which has 5-bit resolution is used to set the current limit for the Boost Converter Stage. The positive reference of DAC2 is set to FVR and the negative reference is set to VSS.

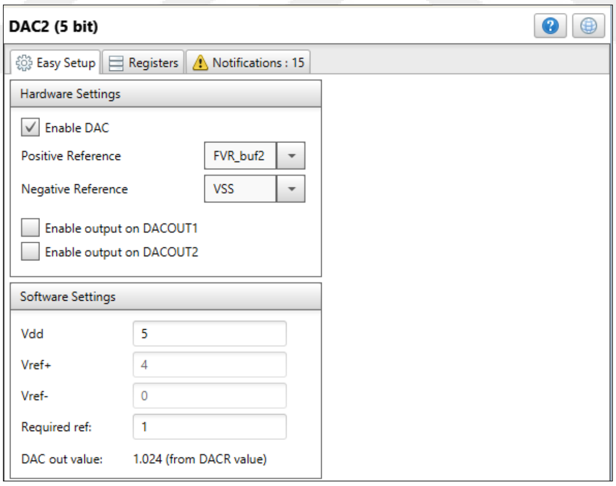


Figure 4.10 DAC2 setting

4.1.6 Operational Amplifier (OPA)

There are two operational amplifiers used in this project, OPA1 and OPA2. OPA1 is used to amplify the Boost Converter stage current sense. The current sense resistor must be a very low resistance since any voltage across this resistor subtracts from the gate drive voltage. The amplified value from OPA1 gives more resolution for the comparator CMP2 to monitor peak current. High Gain Bandwidth Power mode is selected as the power mode and OPA1IN is selected for the Op Amp channel.

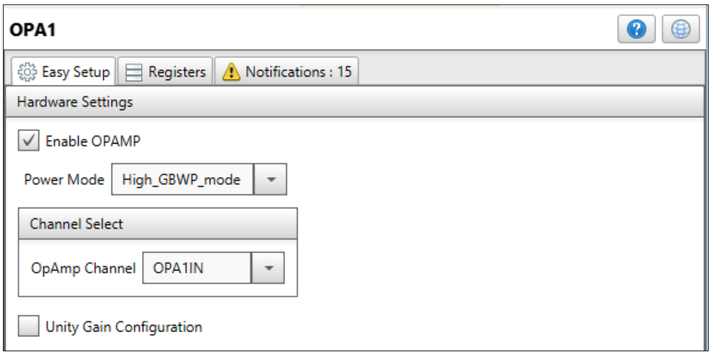


Figure 4.11 OPA1 setting

OPA2 is used to amplify the Signal Generator stage current sense. The setting is similar to OPA1. High Gain Bandwidth Power mode is selected as the power mode and OPA2IN is selected for the Op Amp channel.

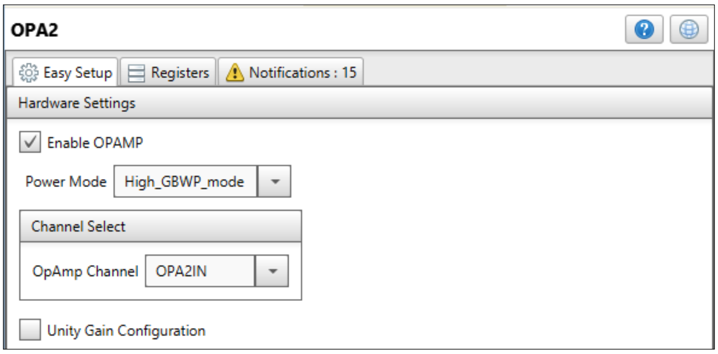


Figure 4.12 OPA2 setting

4.1.7 Comparator (CMP)

Comparators are usually used to interface analog circuits to a digital circuit. By using comparators, two analog voltages are compared and then a digital indication of their relative magnitudes are generated. In this project, CMP1 is used to compare the

Vboost output voltage to the reference voltage selected by DAC1. DAC1 is selected as the positive input of CMP1 and CIN0- is selected as the negative input. The Comparator Zero Latency Filter is enabled in order to prevent the comparator output from oscillating.

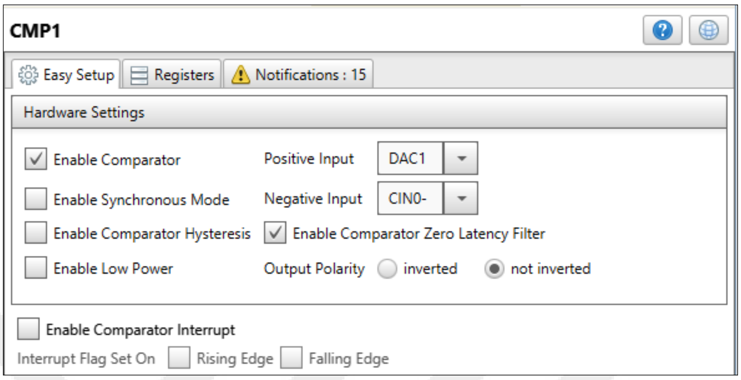


Figure 4.13 CMP1 setting

For CMP2, DAC2 is selected as the positive input and CIN1- as the negative input. The comparator Zero Latency filter is also enabled and the output polarity is set to inverted. CMP2 is used to check if the Boost Converter stage current has reached the target value, and terminate the duty cycle when that happens.

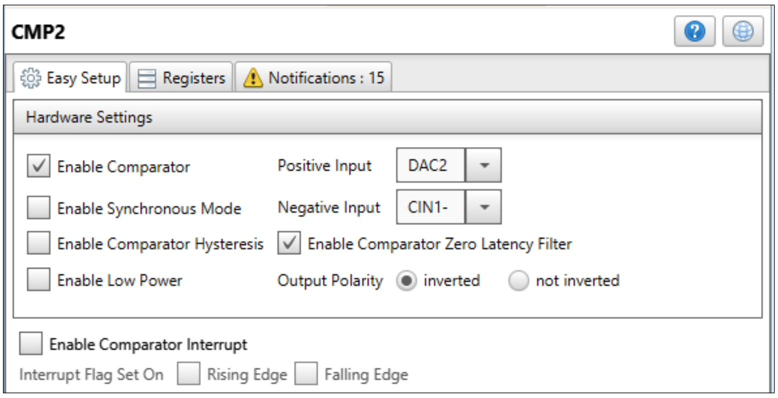


Figure 4.14 CMP2 setting

4.1.8 Configurable Logic Cell (CLC)

The CLC provides programmable logic that operates outside the speed limitations of software execution. The logic cell takes up to 32 input signals and through the use of configurable gates, reduces the 32 inputs to four logic lines that drive one of eight selectable single-output logic functions. In this project, the CLC is used to generate the boost gate driver pulses. In order to do this, CLC1 is set to 1-input D flip-flop with S and R mode. The four gates are set as follows:

1. Gate 1: This gate controls the clock signal to the flip-flop, so the output of PWM3 is selected as the input of this gate. The input is set to inverted input since the falling edge of the PWM signal should be the rising edge of the flip-flop clock.

2. Gate 2: This gate goes into the D input on the flip-flop. This signal comes from Comparator 1 ANDed with a software state "ENABLE". This can be done by selecting C1OUT as the input of this gate and the LC1D2G2T bit as the ENABLE state. All the inputs of this gate are disconnected so that the D input of the flip-flop is low, and effectively the Boost Converter stage is off. Whenever the Boost Converter stage should be turned on, this is done by setting the LC1D2G2T that connects C1OUT to D. This allows the boost pulses to be generated when the Vboost output is less than the threshold.

3. Gate 3: This gate controls the Reset input to the flip-flop. The output of PWM4 is selected as the input of this gate and the input is inverted since the flip-flop will be reset at the falling edge of PWM4. The boost gate drive also will be reset when the Comparator 2 detects that the current target has been reached. Hence, C2OUT is also connected to the input of Gate 3. Gate 3 will

OR the PWM4 and C2OUT signals together and use that value for the Reset input to the flip-flop.

4. Gate 4: This gate controls the Set input of the flip-flop. Since the Set input is not needed, all the inputs of this gate are disconnected.

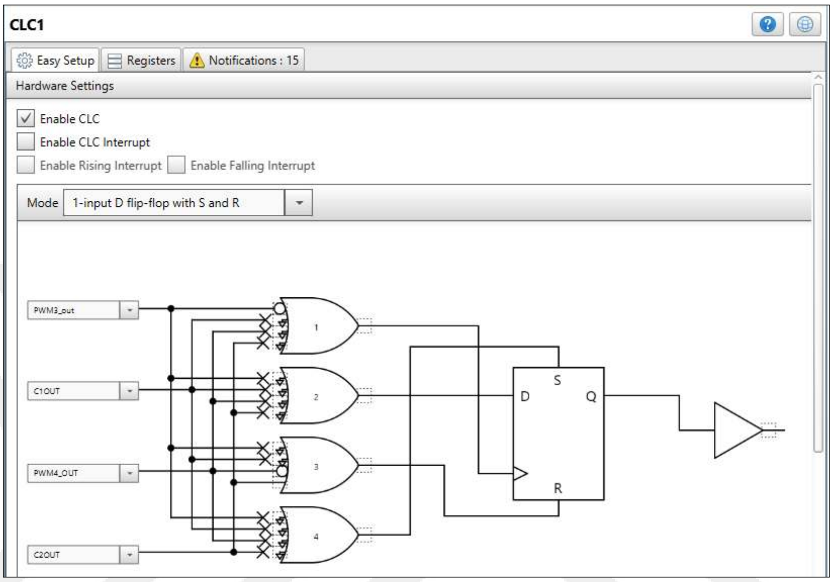


Figure 4.15 CLC1 setting

4.1.9 Numerically Controlled Oscillator (NCO)

NCO is a timer that uses the overflow of an accumulator to generate an output signal. The accumulator overflow is controlled by an adjustable increment value. Hence, the resolution of division does not vary with the limited prescaler/postscaler divider value. This makes NCO the most useful for applications that require frequency accuracy and fine resolution at a fixed duty cycle. In this project, the NCO is used for generating the output frequency for the mesh plate. Fixed Duty Cycle mode is selected as the NCO mode and active_hi is selected as the output polarity. HFINTOSC_16MHz is chosen as the clock source. This is the highest frequency clock available, and it will give the best resolution. The NCO output frequency is specified as 113kHz, which is the nominal operating frequency of the mesh plate. However, this value gets overwritten in software during the frequency sweep. Hence, any value can be put here as the NCO output frequency.

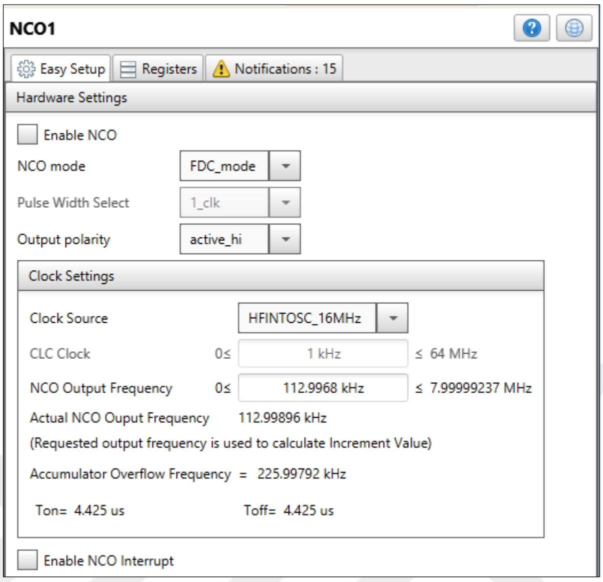


Figure 4.16 NCO1 setting

4.1.10 Analog-to-Digital Converter (ADC)

ADC allows the conversion of an analog input signal to a 10-bit binary representation of that signal. The ADC in this project is used to read the output current, output voltage, and battery voltage. The FOSC/16 is selected as the clock source. This results in a sampling rate of around 87kHz. The positive reference is set to FVR, not VDD. This is because the circuit operates directly from the batteries. If VDD is selected as the positive reference, the value from ADC will vary over the battery voltage range for the same voltage being read. This won't show useful information since it will be difficult to know what the readings mean. For reading the battery voltage, the positive reference is changed to VDD. This is done in the software whenever there is a need to read the battery voltage.

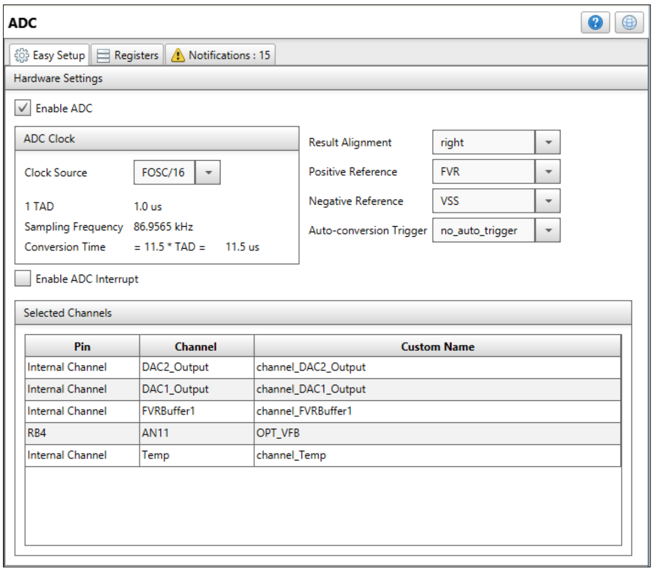


Figure 4.17 ADC setting

4.1.11 Pin Module

In this module, the remaining pins are used for simple input or output. In this part, the remaining pins are selected and the descriptive name is given for each of the pins. The parameters of the pins are also selected whether it is input or output, whether the outputs start high or low, pull-ups for inputs, and Interrupt-On-Change settings.

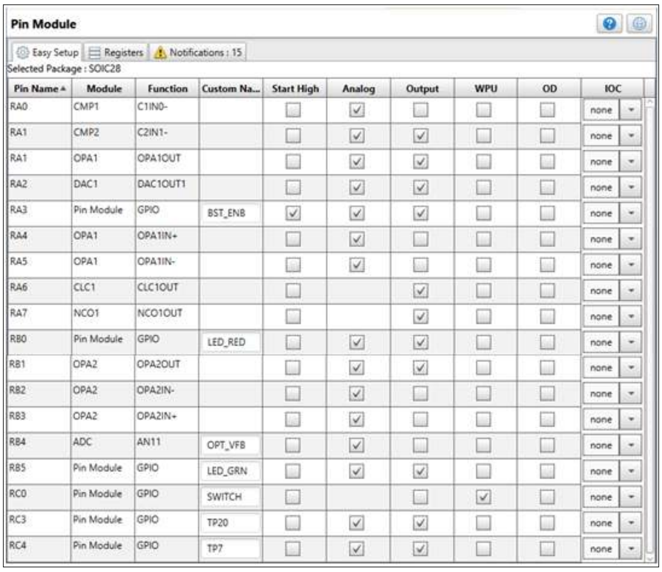


Figure 4.18 Pin Module setting

Figure 4.19 below shows the graphical representation of all the pins assigned to the chip.

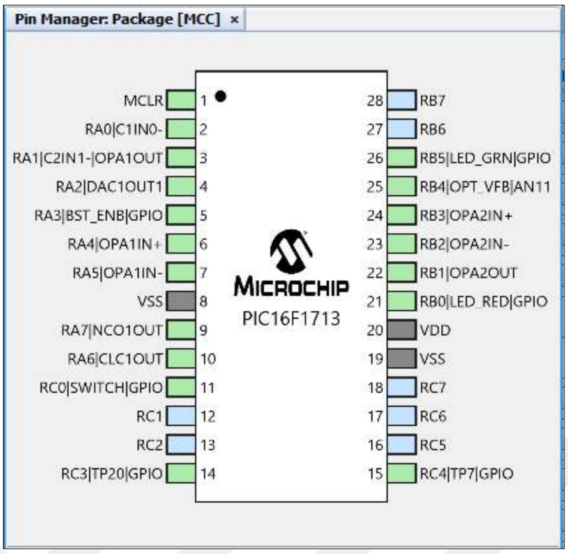


Figure 4.19 Graphical Pin Manager

4.2 CODE IMPLEMENTATION

In the software part, the system is divided into six state machines that have specific tasks and functions that need to be completed in each state respectively. The implementation of state machine makes the documentation easier and more understandable, and breaks the application down into smaller sections that are easily programmed. Figure 4.20 shows the state diagram and the transition between each state.

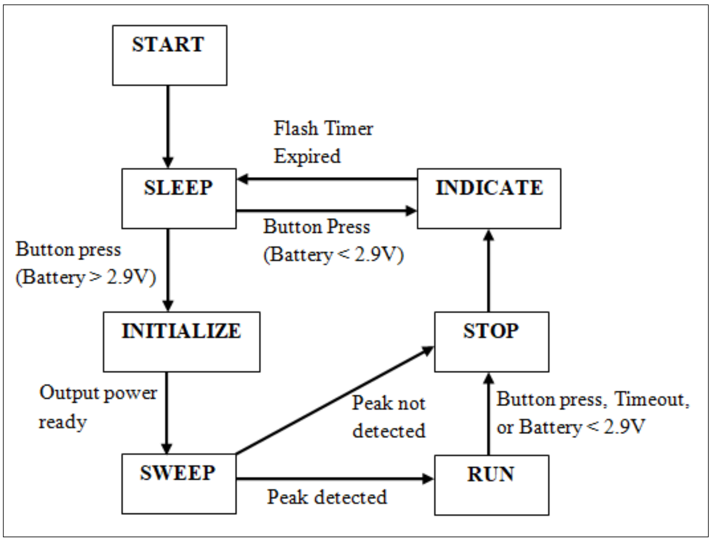


Figure 4.20 State diagram and transition

The system will start when the battery is connected to the circuit. When the battery is connected, the system will enter into the SLEEP state as long as the button is not pressed. This is to ensure that the device is not operated when the user is not ready. When the button is pressed, if the voltage is less than 2.9V, red LED will be flashed and the system will enter into the INDICATE state. If the voltage is greater than 2.9V, the system will enter into the INITIALIZE state. In this state, the initialization of the peripherals and the I/O are done for both BOOST CONVERTER and SIGNAL GENERATOR stages. Then the state moves to the SWEEP state. In the SWEEP state, the frequency sweep runs from 90kHz to 150kHz in order to determine the nominal frequency of the piezoelectric ceramic on the mesh plate. A peak will be detected between low and high frequencies. If the peak is detected, the state will move into the RUN state, whereas if the peak is not detected, the state will move into the STOP state.

In the RUN state, the mesh plate is driven by the generated sine wave signal. The mesh plate will vibrate continuously until it is interrupted by the button, timer, or battery that will make the system to enter into the STOP state. The BOOST CONVERTER and SIGNAL GENERATOR stages are turned off in the STOP state, and the system will run in low power mode and move into the INDICATE state. In the INDICATE state, the LEDs are used as an indicator to show to the user why the device stopped working. Then the system will enter into the SLEEP state which is in this state, the microcontroller enters into the sleep mode and runs with low power until the button is pressed. When the button is pressed, the same process and transitions as explained above will take place.

Figure 4.21 shows the flowchart of the main function of the program. The main function is started by initializing the PIC16F1713 MCU. By using MCC for the configuration and peripherals setup, the initialization of PIC16F1713 is done by just calling the function `SYSTEM_Initialize()`. This makes the code simpler and easier. SLEEP state is assigned as the initial state condition in order to prevent the device from operating when the user is not ready. Then the system will enter into a loop that will check the state condition and executes the process in accordance with the state.

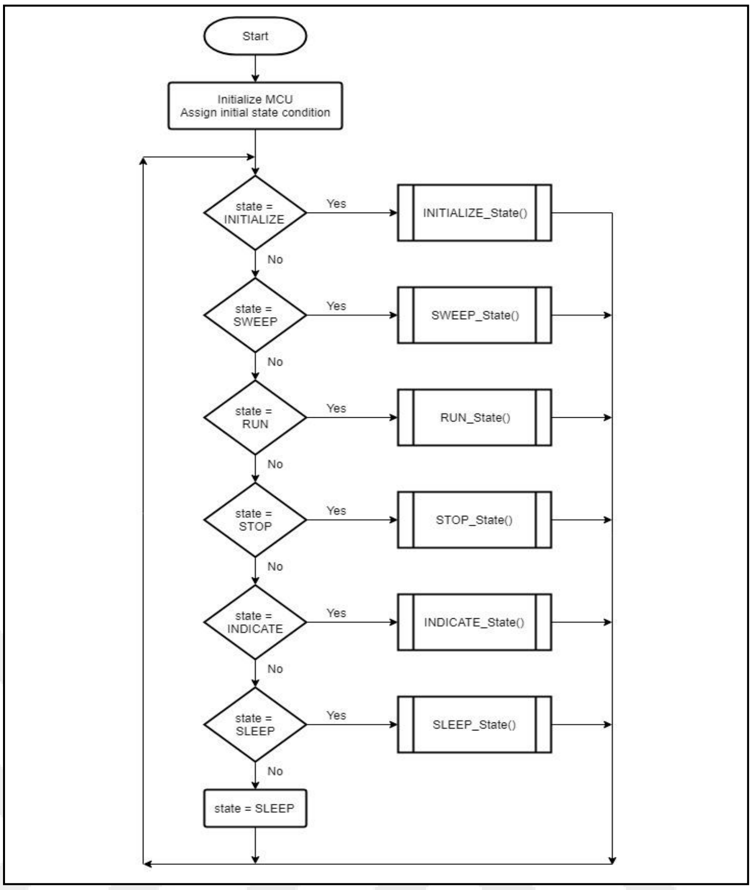


Figure 4.21 Main flowchart

4.2.1 INITIALIZE STATE

INITIALIZE state is started by initializing FVR. FVR is used as a reference for the DAC and ADC modules. Then DAC1 and DAC2 are enabled. DAC1 is used to set the output voltage limit whereas DAC2 is used to set the current limit for the BOOST CONVERTER stage. After setting the frequency to the base value for the frequency sweep, which is 90kHz, the supply voltage and BOOST are enabled by setting the BST_ENB pin low and LC1D2G2T pin high. LC1D2G2T is the D input of Flip Flop in the CLC. By setting this pin high, BOOST CONVERTER stage will start and then a delay is implemented to allow the boost voltage to ramp up and stabilize. When the voltage is stable, which means that the BOOST CONVERTER stage is ready, the SIGNAL GENERATOR stage is enabled by enabling the NCO and initializing it to the lower limit of the frequency sweep, which is 90kHz. Then the state is changed to SWEEP state. The flowchart for this state is shown in Figure 4.22.

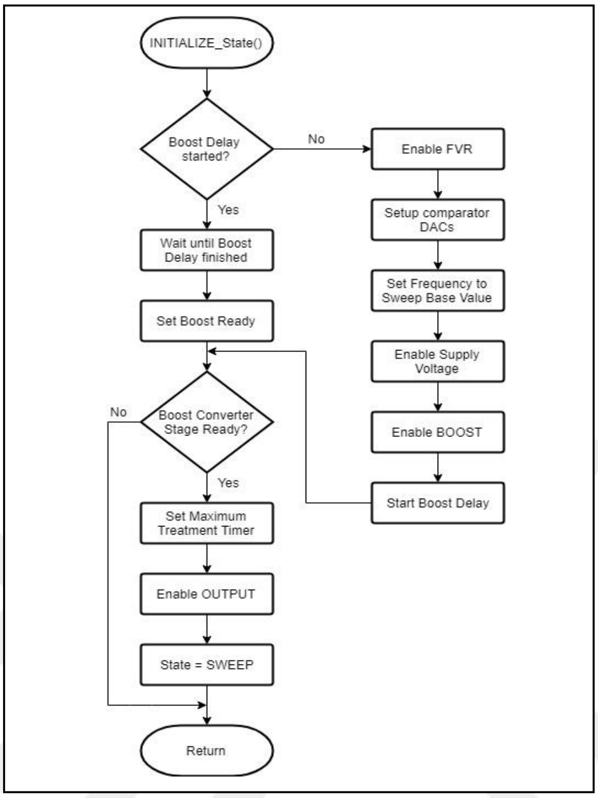


Figure 4.22 INITIALIZE state flowchart

4.2.2 SWEEP STATE

Figure 4.23 shows the flowchart for SWEEP state. In this state, the frequency sweep runs from 90kHz to 150kHz. In each increment of the frequency, the value of the current is sampled and compared with the last sampled current. The value may be decreasing or increasing. In the INITIALIZE state, the current is assigned to the highest possible value before moving to the SWEEP state. Hence, for the peak to be detected, the trend of the current sampled will be decreasing, followed by increasing values, and then followed by decreasing values again. The current will rise when the piezoelectric ceramic approaches resonance.

If the mesh plate is installed and operated properly, the peak will be detected during this process. When the highest value of the current is detected, the corresponding value of current and frequency is saved, and when the trend of the waveform is decreasing again, the frequency is logged as the maximum point. After the upper limit of the frequency has been reached, the frequency recorded as the maximum point is loaded into the NCO and then the state moves into the RUN state. In a case where the peak is not detected, the system will flash red and green LED alternately, and enter into the STOP state.

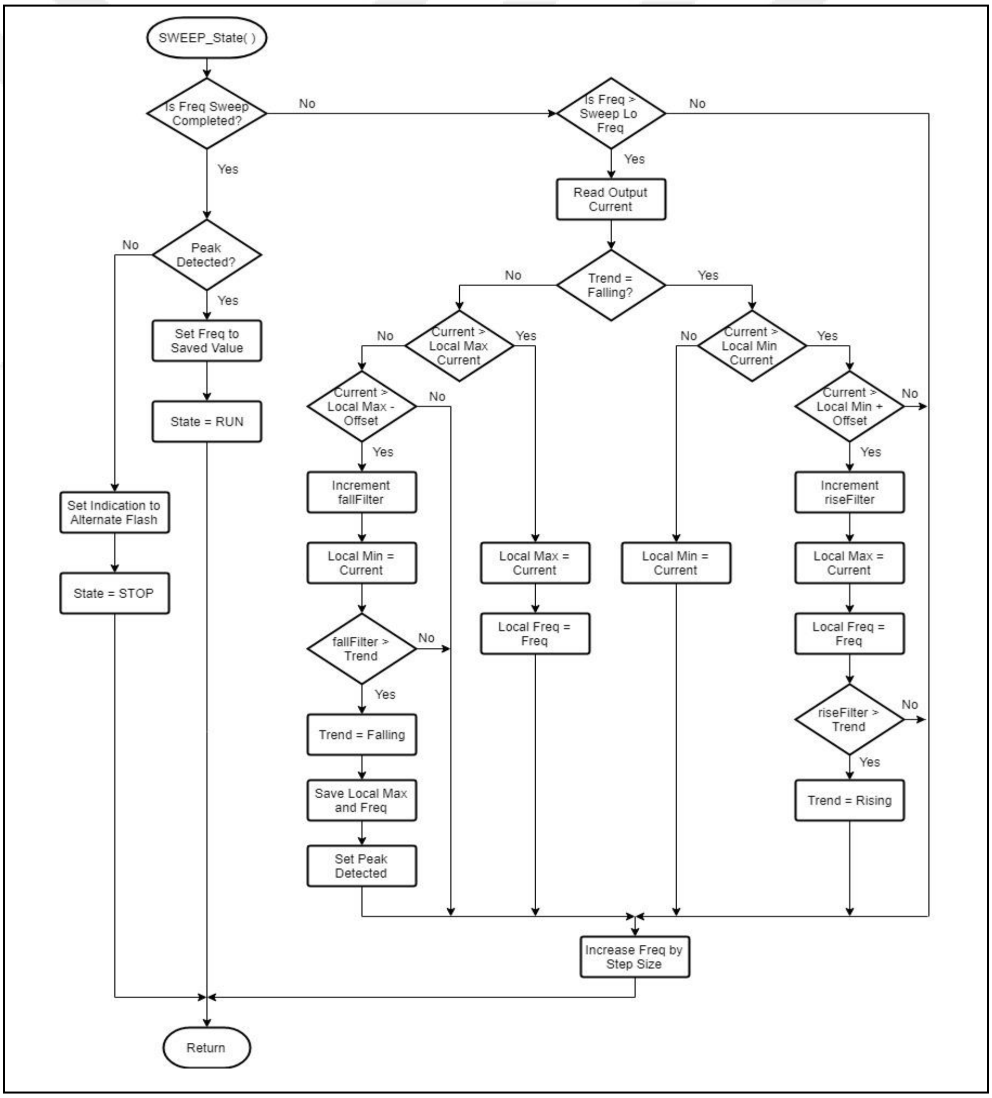


Figure 4.23 SWEEP state flowchart

4.2.3 RUN STATE

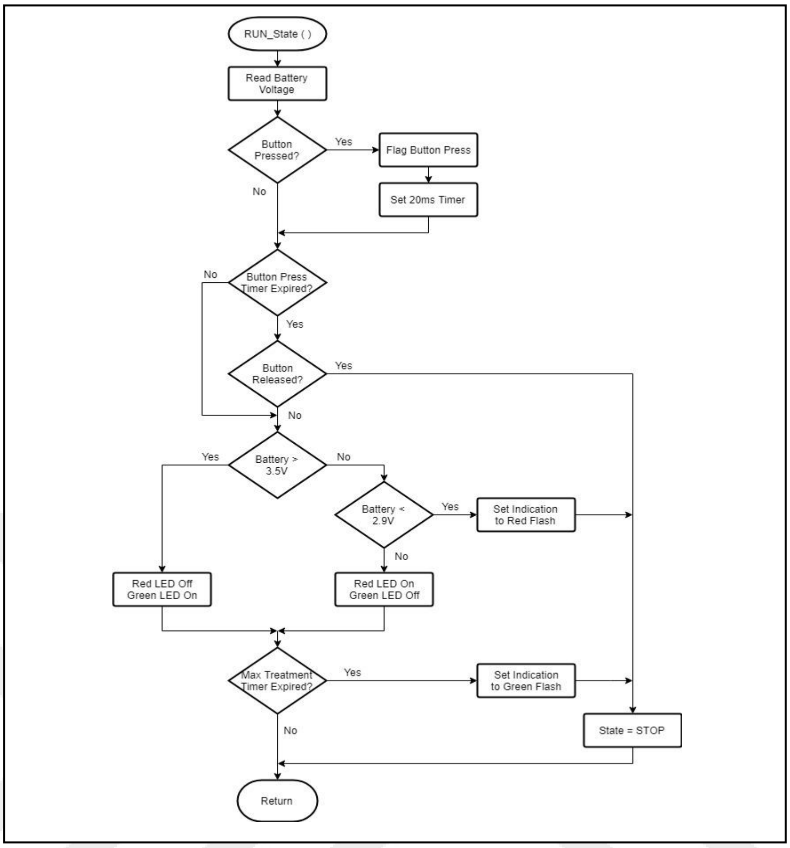


Figure 4.24 RUN state flowchart

In RUN state, the mesh plate runs continuously at the frequency loaded into the NCO until it is interrupted by the button, timer, or battery. The flow of the program coded in this state is shown in Figure 4.24. In this state, the button, the maximum treatment timer counter, and the battery voltage are always monitored. If the button is pressed, the system will enter into STOP state and the output will be off. Hence, the users can shut the nebulizer off at any time they want. At the same time, the maximum treatment timer counter is also monitored. The nebulizer is set to run at a maximum of 15 minutes. If the device operates without stop, after 15 minutes, the device will stop and the green LED will flash. This is to prevent the battery from draining if the device is inadvertently activated by the user, or in a case if the user forgot to shut the device off. The battery voltage is also monitored continuously and the LED is used as an indicator for informing the user about the condition of the battery. If the battery voltage is above 3.5V, the green LED is turned on. This shows that the battery is good and the device should be running smoothly if there are no other problems. When the battery voltage drops to below 3.5V, red LED will be turned on to indicate that the battery is low. Then, if the battery voltage falls below 2.9V, the system will flash red LED and enter into STOP state.

4.2.4 STOP STATE

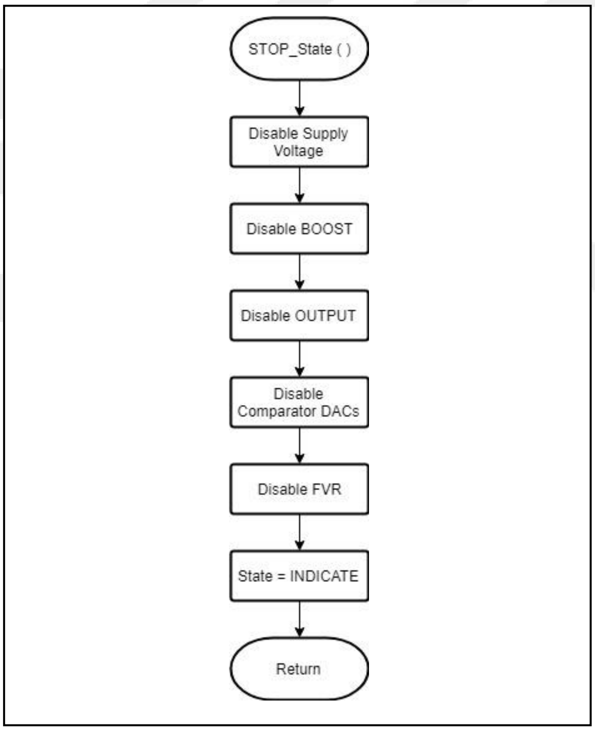


Figure 4.25 STOP state flowchart

When the system into the STOP state, the supply voltage is cut off from the circuit. This is done by setting the BST_ENB pin high. Then the CLC and NCO outputs are also driven low. After the DACs and FVR are turned off as a preparation for low power sleep, the system enters into the INDICATE state. Figure 4.25 above shows the flowchart used for programming the STOP state.

4.2.5 INDICATE STATE

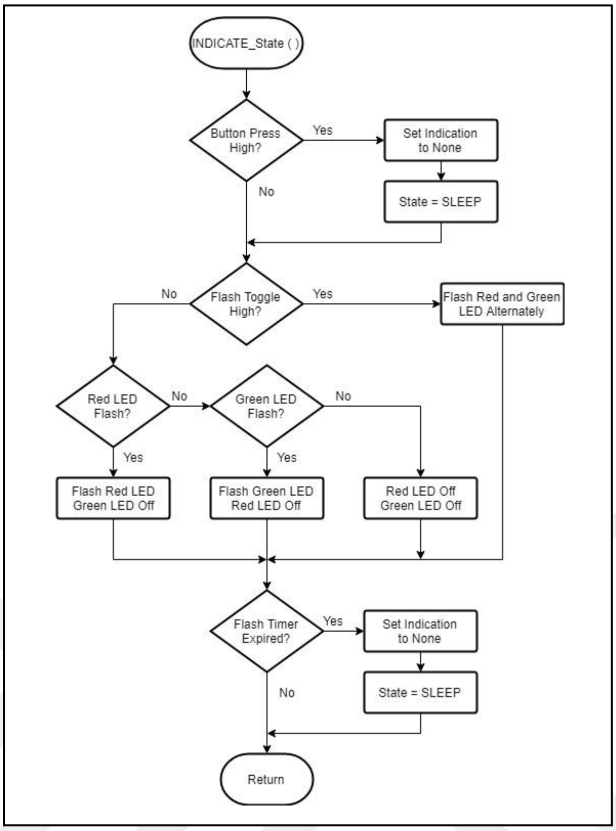


Figure 4.26 INDICATE state flowchart

The INDICATE state controls the output of LED in order to inform the user about the reason of the device shutting down. As shown in Figure 4.26, this state is started by checking the Button Press Flag. If it is high, then the indication will be set to none (LEDs will be turned off) and the state will move into the SLEEP state. This is, for example, in a condition where the user pressed the button during the nebulizer is running (the system is in RUN state). When the button is pressed, the Button Press Flag will be high and the device will stop. The LEDs will be turned off and the device enters into sleep mode. If the Button Press Flag is low, then the Flash Toggle Flag will be checked. If the Flash Toggle Flag is high, the red and green LED will be flashed alternately. This is when no peak is detected during the frequency sweep. Then the flags for flashing red LED and green LED are checked. If it is high, then the respective LED will be flashed. If both of the flags are low, both of LEDs will be turned off and the state will move into the SLEEP state.

4.2.6 SLEEP STATE

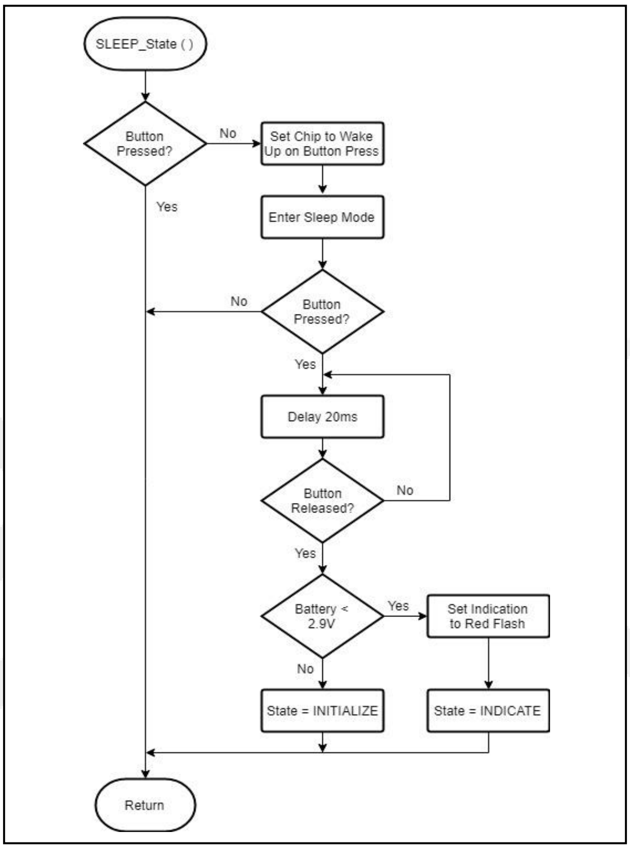


Figure 4.27 SLEEP state flowchart

Figure 4.27 shows the flowchart of SLEEP state that is programmed for the nebulizer developed in this study. The system should be in low power condition when the program enters into the SLEEP state since the power is already cut off from driving the mesh plate in the STOP state. The chip is set to wake up on a button press. However, there might be a possibility that the chip is awakened by some other event although it has been set to wake on a button press. In order to make sure that the chip will wake up only on button press, the button is tested whether it is pressed or not. If the button is not pressed, then the SLEEP state will be re-entered and the chip will be put back in the Sleep mode. There might be also a situation where the button is long pressed and we didn't want the chip to wake up before the button is released, so the button is checked whether it has been released or not. If it is still pressed, the code will keep looping until it is released. When the button is released, the battery voltage is checked. If it is below than 2.9V, red LED will be flashed and the state moves into the INDICATE state. If the battery is higher than 2.9V, then the system will enter into the INITIALIZE state and starts the nebulizer.

CHAPTER 5

RESULT AND DISCUSSION

Figure 5.1 shows the experimental setup of vibrating mesh nebulizer circuit constructed on the breadboard. The circuit is powered by using three AAA batteries, supplying voltage of about 4.5V. From the figure, it can be seen that the aerosol has been successfully generated. The green LED lights up showing that the circuit is working properly and the output is generated.

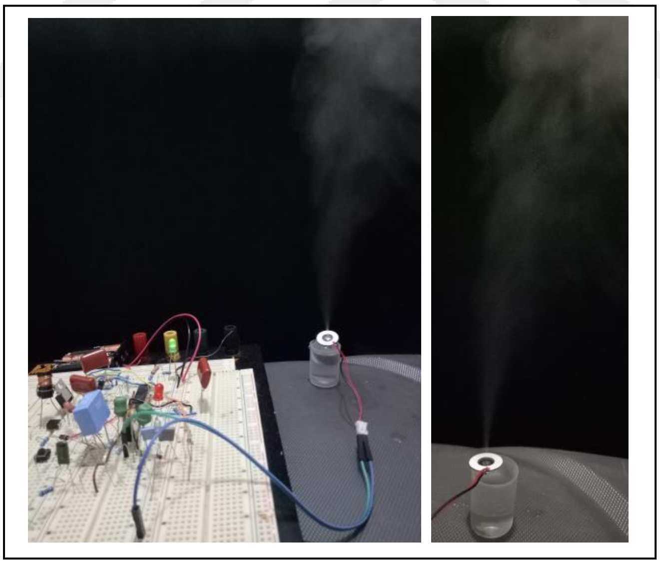


Figure 5.1 Experimental setup

The sine wave signal generated for driving the micropore mesh plate is shown in Figure 5.2. The upper waveform is the output generated from the circuit whereas the second waveform is the output of NCO module that is used to drive the MOSFET in the Signal Generator Stage circuit. It can be seen also from the figure that the sine wave signal is generated at the frequency of 113.5kHz, which is the nominal frequency for the mesh plate.

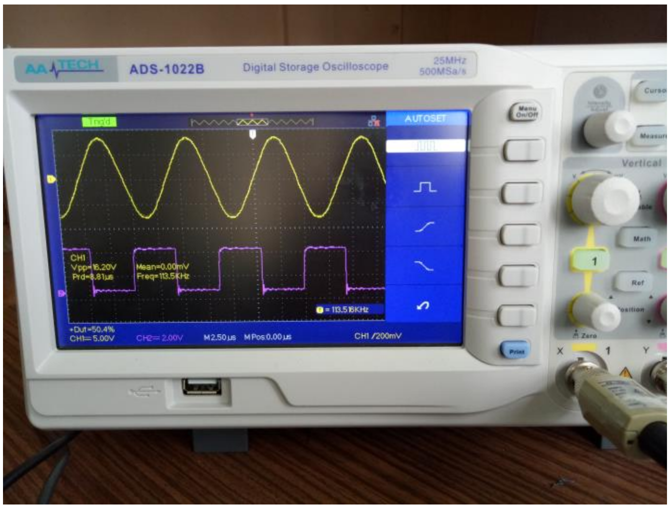


Figure 5.2 Output waveform

In this circuit, LED has been used as an indicator to inform the user about the possible problem of the device. Table 5.1 shows the LED conditions and the explanation of the device conditions.

Table 5.1 LED output indicator

LED Condition	Device Condition
Green LED turned on	The device is working properly
Red LED turned on	Battery is low. The device is working with low power condition.
Green and red LED flashes alternately	No peak is detected during frequency sweep. No output is generated.
Green LED flash	Maximum treatment time has been reached. The device is stopped working.
Red LED flash	Battery is very low. The device is stopped working.

There are many advantages of vibrating mesh nebulizer compared to jet and ultrasonic nebulizers. This new type of nebulizer can improve the effectiveness of nebulizer treatment. The short treatment time and the silent operation of the device can be a decisive factor for the patient to adhere to the treatment. Table 5.2 shows the comparison between jet, ultrasonic, and mesh nebulizer.

Table 5.2 Comparison between jet, ultrasonic and mesh nebulizer

	Jet	**Ultrasonic**	**Mesh**
Size	Big	Medium	Small
Power Source	Compressed gas or electrical mains	Electrical mains	Batteries or electrical mains
Portability	Restricted	Restricted	Portable
Output Rate	Low	Higher	Highest
Treatment Time	Long	Intermediate	Short
Residual Volume	0.8 - 2.0mL	Variable but low	Less than 0.2mL

From the table above, we can see that mesh nebulizer has a better performance compared to jet and ultrasonic nebulizer. Since mesh nebulizer can be powered up by using batteries, it is portable. The small size of mesh nebulizer also makes it easier to be carried everywhere. The output rate of mesh nebulizer is also the highest among these three types of nebulizer. Hence, higher dosage of medicine can be delivered to the patient in a shorter treatment time. The higher the output rate, the shorter the treatment time. The residual volume of mesh nebulizer is also very low compared to jet and ultrasonic nebulizer.

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

6.1 CONCLUSION

In the past few decades, the medical devices technology industry has been expanding rapidly. The advancement of medical devices technology has been a great help for medical and health practitioners in improving human's health. Vibrating mesh nebulizer is one of the medical devices that has been studied and developed in these past few years. Hence, the vibrating mesh nebulizer design is also studied and developed in this research. In this study, the vibrating mesh nebulizer design by using PIC16F1713 as the microcontroller, and micropore mesh plate has been developed successfully. The aerosol has been generated from the micropore mesh plate, which the mesh plate is driven by the sine wave signal generated from the circuit. The vibrating mesh nebulizer design developed in this study has advantages of compact design, portable, silent operation, user-friendly, and works using low frequency and low voltage. The compact and portability features make it easier for the user to use it during the nebulization therapy.

The circuit is also designed with flexibility in terms of frequency due to the frequency sweep. Hence, the circuit also can be used for driving other mesh plates operating at frequencies ranging between 100-150kHz. This makes it suitable for other applications in aerosol and mist production device as well.

6.2 RECOMMENDATIONS FOR FUTURE WORKS

The nebulizer market is expected to grow at a rate of 5% in the next 10 years. Hence, more research and study should be done in the nebulizer device area in order to design more efficient and more convenient nebulizer device. For more efficient nebulizing therapy, there are some improvements can be made from the vibrating mesh nebulizer design developed in this study:

1. The nebulizer can be designed to generate aerosol only during the inhalation period of the patient. This will increase the percentage of aerosol inhaled by the lung and decrease the aerosol wasted to the air.

2. The nebulizer can be designed to cater specific type of drug or specific disease. The specific drug-device combination will allow the device to optimized in accordance with the drug dosage pack. This can reduce the cost and treatment time for the patient.

3. A system that will monitor the usage of the device can also be integrated into the device. The system will record each time the device is used for nebulizing therapy. This will make it easier for the doctors to monitor and plan the treatment for the patients.

REFERENCES

[1] Muers, M. F. (1997). Overview of nebuliser treatment. *Thorax*, **52(Suppl 2)**, S25.

[2] Anderson, P. J. (2005). History of aerosol therapy: liquid nebulization to MDIs to DPIs. *Respiratory Care*, **50(9)**, 1139-1150.

[3] O'callaghan, C., Barry, P. W. (1997). The science of nebulised drug delivery. *Thorax*, **52(Suppl 2)**, S31.

[4] Nerbrink, O., Dahlback, M., Hansson, H. C. (1994). Why do medical nebulisers differ in their output and particle size characteristics?. *Journal of Aerosol Medicine*, **7(3)**, 259-276.

[5] Hess, D., Fisher, D., Williams, P., Pooler, S., Kacmarek, R. M. (1996). Medication nebulizer performance: effects of diluent volume, nebulizer flow, and nebulizer brand. *Chest*, **110(2)**, 498-505.

[6] Smith, E. C., Denyer, J., Kendrick, A. H. (1995). Comparison of twenty three nebulizer/compressor combinations for domiciliary use. *European Respiratory Journal*, **8(7)**, 1214-1221.

[7] Ari, A. (2014). Jet, ultrasonic, and mesh nebulizers: an evaluation of nebulizers for better clinical outcomes. *Eurasian Journal of Pulmonology*, **16**, 1-7.

[8] Gardenhire, D. S., Ari, A., Hess, D., Myers, T. R. A Guide to Aerosol Delivery Devices for Respiratory Therapists. (2013). Dallas, TX: American

Association for Respiratory Care. Available at: https://www.aarc.org/wp-content/uploads/2015/04/aerosol_guide_rt.pdf

[9] Yeo, L. Y., Friend, J. R., McIntosh, M. P., Meeusen, E. N., & Morton, D. A. (2010). Ultrasonic nebulization platforms for pulmonary drug delivery. *Expert Opinion on Drug Delivery*, **7(6)**, 663-679.

[10] Vecellio, L. (2006). The mesh nebuliser: a recent technical innovation for aerosol delivery. *Breathe*, **2(3)**, 252-260.

[11] Yardimci, A. (2011). Microcontroller based jet nebulizer design with ANFIS compessor control for domiciliary use. In *Innovations in Intelligent Systems and Applications (INISTA), 2011 International Symposium on* (pp. 428-431). IEEE.

[12] Tsai, C. S., Mao, R. W., Lin, S. K., Zhu, Y., Tsai, S. C., Brenner, M., Mahon, S., Mukai, D., & Boss, G. (2013). Pocket-sized ultrasonic nebulizer for inhalation drug delivery. In *Ultrasonics Symposium (IUS), 2013 IEEE International* (pp. 1190-1192). IEEE.

[13] Staderini, E. M., & Rusu, R. (2014). Pressure activated synchronizer of nebulizing therapy in BPAP artificial ventilation. In *Medical Information and Communication Technology (ISMICT), 2014 8th International Symposium on* (pp. 1-3). IEEE.

[14] Ivanova, M. D., & Glazova, A. Y. (2015). Nebulizer improvement for children suffering from bronchial asthma. In *Young Researchers in Electrical and Electronic Engineering Conference (ElConRusNW), 2015 IEEE NW Russia* (pp. 329-331). IEEE.

[15] Jeng, Y. R., Su, C. C., Feng, G. H., & Peng, Y. Y. (2007). An investigation into a piezoelectrically actuated nebulizer with μEDM-made micronozzle array. *Experimental Thermal and Fluid Science*, **31(8)**, 1147-1156.

[16] Shen, S. C., Wang, Y. J., & Chen, Y. Y. (2008). Design and fabrication of medical micro-nebulizer. *Sensors and Actuators A: Physical*, **144(1)**, 135-143.

[17] Lin, C. Y., Meng, H. C., & Fu, C. (2011). An ultrasonic aerosol therapy nebulizer using electroformed palladium-nickel alloy nozzle plates. *Sensors and Actuators A: Physical*, **169(1)**, 187-193.

[18] Kumar, G. S., & Indira, S. (2014). Embedded boost converter using voltage feedback technique. *International Journal of Research in Engineering and Technology*, **2(2)**, 207-212.

[19] Hawkins, B., & Feng, Z. Vibrating Mesh Nebulizer Reference Design. (2017). Microchip Technology Incorporated. Available at: http://ww1.microchip.com/downloads/en/AppNotes/00002265B.pdf

[20] Ellison, C. Designing a boost-switching regulator with the MCP1650. (2005). Microchip Technology Incorporated. Available at: http://ww1.microchip.com/downloads/en/AppNotes/00980a.pdf

[21] Shen, S. C., Liou, S. N., Wang, Y. R., Chien, C. P., Chen, Y. C., Yen, C. S., Pan, I. K., Lee, C. J., Yang, C. W., & Wang, M. R. (2004). The design and manufacturing of new micro nebulizer. In *Computer Architectures for Machine Perception, 2003 IEEE International Workshop on* (pp. 103-106). IEEE.

[22] Chetan, M., & Negoias, A. (2011). New approaches to nebulizer drug delivery. In *Advanced Topics in Electrical Engineering (ATEE), 2011 7th International Symposium on* (pp. 1-4). IEEE.

[23] Dolovich, M. B., & Dhand, R. (2011). Aerosol drug delivery: developments in device design and clinical use. *The Lancet*, **377(9770)**, 1032-1045.

APPENDIX

This appendix contains MPLAB X IDE codes that were used in this project.

1. Header File

```
#ifndef NEBULIZER_H
#define    NEBULIZER_H

#ifdef     __cplusplus
extern "C" {
#endif

#pragma warning disable 520      // function is never called
#pragma warning disable 751      // arithmetic overflow in
constant expression
#pragma warning disable 752      // conversion to shorter data
type

/*
 * Project Defines
 */

// State Machine
typedef enum
{
    INIT,
    SWEEP,
    RUN,
    STOP,
    INDICATE,
    SLEEP
}STATE;

typedef union
{
    uint16_t all;
    struct
    {
        unsigned SoftStart:1;          // Soft Start Active
        unsigned BoostDelay:1;         // Boost Delay Active
        unsigned BoostReady:1;         // Boost Stage Ready
        unsigned PeakDetect:1;         // Sweep Function
Detected a Local Maximum Peak
        unsigned Falling:1;            // Sweep Current
Decreasing
        unsigned ButtonPress:1;        // Button Press Detected
```

```
        unsigned Debounce:1;          // Button Debounce Timer
        unsigned FlashToggle:1;       // Set if LED Red or
Green Should Flash Alternately
        unsigned FlashRed:1;          // Set if Red LED Should
Flash
        unsigned FlashGrn:1;          // Set if Green LED
Should Flash
        unsigned FlashTimer:1;        // Set When Flash Timer
Expires
        unsigned bitB:1;        //
        unsigned bitC:1;        //
        unsigned bitD:1;        //
        unsigned bitE:1;        //
        unsigned bitF:1;        //
    }bits;
}FLAGWORD;

extern STATE state;
extern FLAGWORD flags;

extern uint16_t maxTreatmentTimer;


// INIT State

#define POWER       (BST_ENB_LAT)       // Pin to Enable Power
Circuit
#define POWER_ON    POWER = 0
#define POWER_OFF   POWER = 1

#define BOOST_ON    CLC1GLS1 = 8    // LC1D2G2T = 1
#define BOOST_OFF   CLC1GLS1 = 0    // LC1D2G2T = 0
#define BOOST_VOLTAGE   10

#define OUTPUT_ON   NCO1CONbits.N1EN = 1
#define OUTPUT_OFF  NCO1CONbits.N1EN = 0
#define OUTPUT_FREQ(x) {                            \
                        NCO1INCH = (uint8_t)((x)>>8);      \
                        NCO1INCL = (uint8_t)((x)&0xff);    \
                       }

#define TIMER1MS(x)  (uint16_t)(65536-((x)*500000))

#define MS100   TIMER1MS(100)
#define MS1000  TIMER1MS(1000)

#define PERIOD        80                           // PR2 Value
#define SOFTSTART    (unsigned)(0.50 * PERIOD)     // 50% Duty
Cycle
#define REDUCEDPOWER (unsigned)(0.75 * PERIOD)     // 75% Duty
Cycle
#define FULLPOWER    (unsigned)(0.92 * PERIOD)     // 85% Duty
Cycle

#define MINUTE          120                  // 1 Minute
```

```
#define MAX_OP          (15 * MINUTE)       // Maximum
Operation Time


// SWEEP State

#define HI_FREQ 150000          // High Value for Frequency
Scan
#define LO_FREQ 100000          // Low Value for Frequency
Scan
#define BS_FREQ 90000           // Base Frequency
#define NOISELEVEL 90           // ADC Counts Necessary to
Switch Between Increasing/!Increasing
#define TREND 130                // Number of Samples for
Rising/Falling Before Change of Direction is Accepted
#define FREQ_STEP   1

#define FREQ_OUT    (LATA7)
#define FREQ_PIN    (RA7)

extern uint16_t outCurrent;     // Output Current
extern uint16_t localMin;       // Local Minimum Output
Current
extern uint16_t localMax;       // Local Maximum Output
Current
extern uint16_t maxVal;         // Detected Peak Output
Current
extern uint16_t localFreq;      // Frequency at Local Maximum
extern uint16_t maxFreq;        // Detected Peak Frequency
extern uint16_t newFreq;        // New Frequency to Write to
NCO
extern uint8_t  riseFilter;     // Rising Trend Filter
extern uint8_t  fallFilter;     // Falling Trend Filter

#define OPA2OUT 0xA     //channel AN10 = OPA2OUT


// RUN State

#define LOW_BATTERY 3.5         // Low Battery Detect Voltage
#define HYST 0.05               // Hysteresis in Low Battery
Reading
#define SHUTDOWN 2.9            // Low Battery Shutdown

#define ADC_Ref_VDD()   ADCON1bits.ADPREF = 0
#define ADC_Ref_FVR()   ADCON1bits.ADPREF = 3

#define MS20    TIMER1MS(20)

#define RED_LED_ON()    LED_RED_SetHigh()
#define RED_LED_OFF()   LED_RED_SetLow()

#define GRN_LED_ON()    LED_GRN_SetHigh()
#define GRN_LED_OFF()   LED_GRN_SetLow()

extern adc_result_t batteryVoltage; // Battery Voltage Reading
```

```
// INDICATE State

extern uint8_t TimerValue;
extern uint8_t FlashValue;


// STOP State

#define FVR_OFF()   FVRCONbits.FVREN = 0
#define DAC1_OFF()  DAC1CON0bits.DAC1EN = 0
#define DAC2_OFF()  DAC2CON0bits.DAC2EN = 0


//--------------------------------------------------------------
-------
// These #defines are for calculations based on definitions
above and should not require any changes
//
//--------------------------------------------------------------
-------

#define NCORATE 7.62939453      // Set by Clocking into NCO -
Number of Hz per step
#define _XTAL_FREQ  16000000    // Operating Frequency

#define FREQ_BASE   (uint16_t)(BS_FREQ/NCORATE)
#define FREQ_START  (uint16_t)(LO_FREQ/NCORATE)
#define FREQ_TOP    (uint16_t)(HI_FREQ/NCORATE)

#define ADC_VREF  2.048
#define LOW_BATTERY_P_HYST ((int)((ADC_VREF *
1023)/(LOW_BATTERY + HYST)))
#define LOW_BATTERY_M_HYST ((int)((ADC_VREF *
1023)/(LOW_BATTERY - HYST)))

#define SHUTDOWN_P_HYST ((int)((ADC_VREF * 1023)/(SHUTDOWN +
HYST)))
#define SHUTDOWN_M_HYST ((int)((ADC_VREF * 1023)/(SHUTDOWN -
HYST)))

#define CMP1_VREF 2.048
#define CMP1_STEP ((CMP1_VREF)/256)
#define BV_DIVIDER ((15.0)/(221.0 + 15.0))            // R5,
R8
#define BOOST_CMP  ((uint8_t)((BOOST_VOLTAGE *
BV_DIVIDER)/CMP1_STEP))


//--------------------------------------------------------------
-------
// Function Prototypes
//
//--------------------------------------------------------------
-------
```

```
// State Functions

extern void INITIALIZE_State(void);
extern void SWEEP_State(void);
extern void RUN_State(void);
extern void STOP_State(void);
extern void INDICATE_State(void);
extern void SLEEP_State(void);

// Timer Function
void setTimer(uint16_t value);


#ifdef     __cplusplus
}
#endif

#endif     /* NEBULIZER_H */
```

2. Main Source File

```
#include "mcc_generated_files/mcc.h"
#include "nebulizer.h"

void main(void)
{
    // initialize the device
    SYSTEM_Initialize();

    state = SLEEP;

    while(1)
    {
        switch(state)
        {
            case INIT:
                INITIALIZE_State();
                break;

            case SWEEP:
                SWEEP_State();
                break;

            case RUN:
                RUN_State();
                break;

            case STOP:
                STOP_State();
                break;

            case INDICATE:
```

```
                INDICATE_State();
                break;

            case SLEEP:
                SLEEP_State();
                break;

            default:
                state = SLEEP;
        }
    }

    return;
}

void setTimer(uint16_t timerValue)
{
    // Write the TMR1 register
    TMR1_WriteTimer(timerValue);

    //Start Timer1
    TMR1_StartTimer();

    // Wait for Timer 1 to overflow
    while(!TMR1_HasOverflowOccured());

    // Clear the TMR1 interrupt flag
    TMR1IF = 0;

    // Reload the initial value of TMR1
    TMR1_Reload();

}
```

3. Initialize State Codes

```
#include "mcc_generated_files/mcc.h"
#include "nebulizer.h"


// Global RAM
STATE state;
FLAGWORD flags;
uint16_t maxTreatmentTimer;

uint16_t outCurrent;      // Output Current
uint16_t localMin;        // Local Minimum Output Current
uint16_t localMax;        // Local Maximum Output Current
uint16_t maxVal;          // Detected Peak Output Current
uint16_t localFreq;       // Frequency at Local Maximum
uint16_t maxFreq;         // Detected Peak Frequency
uint16_t newFreq;         // New Frequency to Write to NCO
uint8_t  riseFilter;      // Rising Trend Filter
```

```
uint8_t  fallFilter;     // Falling Trend Filter


void INITIALIZE_State(void)
{
    if (flags.bits.BoostDelay == 0)
    {
        PWM4_LoadDutyValue(SOFTSTART);
        //* Enable FVR
        FVR_Initialize();

        // Set up Comparator DACs
        DAC1_Initialize();                // Enable DAC1
        DAC1_SetOutput(BOOST_CMP);        // Sets Boost Voltage

        DAC2_Initialize();                // Enable DAC2
        DAC2_SetOutput(0x0a);             // Set for Max Current

        //* Set Frequency to Sweep Base Value
        OUTPUT_FREQ(FREQ_BASE);

        //* Enable Supply Voltage
        POWER_ON;

        //* Enable BOOST
        BOOST_ON;

        //* Set Boost Delay
        setTimer(MS100);                  // Delay for Boost
Stage Startup
        flags.bits.BoostDelay = 1;        // Start Timer
    }
    else
    {
        if(TMR1IF)
        {
            //TMR1_StopTimer();
            flags.bits.BoostReady = 1;
            TMR1IF = 0;
        }
    }
    if (flags.bits.BoostReady)
    {

        PWM4_LoadDutyValue(FULLPOWER);

        //* Set Max Treatment Timer
        maxTreatmentTimer = MAX_OP;       // Set up Maximum
Treatment Time

        flags.bits.PeakDetect = 0;        // No Peak Detected
        maxVal = 0;
        maxFreq = FREQ_START;             // Lowest Valid
Frequency
        localMax = 0;                     // Set to Lowest
Possible
```

```
        localMin = 1023;                // Set to Highest
Possible
        riseFilter = 0;                 // Clear Filters
        fallFilter = 0;                 // Clear Filters
        flags.bits.Falling = 1;         // Assume Value is
Decreasing

        //* Enable Output
        OUTPUT_ON;

        //* State = SWEEP
        state = SWEEP;
        TP20_LAT = 1;
        ADC_Ref_FVR();                  // Set ADC Reference
to FVR for Freq Sweep

        flags.bits.BoostDelay = 0;
        flags.bits.BoostReady = 1;
    }
}
```

4. Sweep State Codes

```
#include "mcc_generated_files/mcc.h"
#include "nebulizer.h"


// Global RAM

uint16_t outCurrent;     // Output Current
uint16_t localMin;       // Local Minimum Output Current
uint16_t localMax;       // Local Maximum Output Current
uint16_t maxVal;         // Detected Peak Output Current
uint16_t localFreq;      // Frequency at Local Maximum
uint16_t maxFreq;        // Detected Peak Frequency
uint16_t newFreq;        // New Frequency to Write to NCO
uint8_t  riseFilter;     // Rising Trend Filter
uint8_t  fallFilter;     // Falling Trend Filter


void SWEEP_State(void)
{
    if (NCO1INC > FREQ_TOP)         // Frequency Sweep
Completed?
    {
        TP20_LAT = 0;
        ADC_Ref_VDD();          // Set ADC to VDD to read
Battery Voltage

        if (flags.bits.PeakDetect)
        {
            //* Set Freq to Saved Value
            OUTPUT_FREQ(maxFreq);
```

```
            //* Enter RUN State
            state = RUN;
        }
        else
        {
            //* Set Indication to Alternate LED Flash
            flags.bits.FlashToggle = 1;
            flags.bits.FlashRed = 1;
            flags.bits.FlashGrn = 1;

            RED_LED_ON();


            //* Enter STOP State
            state = STOP;
        }
    }
    else                            // Frequency Sweeping
    {
        TP7_LAT = 0;

        if (NCO1INC > FREQ_START)
        {
            //* Read Output Current
            outCurrent = ADC_GetConversion(OPA2OUT);

            //* Trend == Falling?
            if (flags.bits.Falling)
            {
                TP20_LAT = 1;
                //* Current > Local Min Current?
                if (outCurrent > localMin)
                {
                    //* Current > Local Min + Offset?
                    if (outCurrent > (localMin + NOISELEVEL))
                    {
                        riseFilter++;
                        localMax = outCurrent;
                        localFreq = NCO1INC;

                        if (riseFilter > TREND)
                        {
                            //* Trend = Rising
                            flags.bits.Falling = 0;
                        }
                    }
                }
                else
                {
                    //* Local Min = Current
                    localMin = outCurrent;

                    if (riseFilter)
                    {
```

```
                        riseFilter = 0;
                    }
                }
            }
            else
            {
                TP20_LAT = 0;
                //Current > Local Max Current?
                if (outCurrent > localMax)
                {
                    //* Local Max = Current
                    localMax = outCurrent;
                    //* Local Freq = Freq
                    localFreq = NCO1INC;

                    if (fallFilter)
                    {
                        fallFilter = 0;
                    }
                }
                else
                {
                    //* Current < Local Max Current - Offset?
                    if (outCurrent < (localMax - NOISELEVEL))
                    {
                        fallFilter++;
                        localMin = outCurrent;

                        if (fallFilter > TREND)
                        {
                            //* Trend = Falling
                            flags.bits.Falling = 1;
                            //* Save Local Max and Freq
                            if ((maxVal < localMax) &&
(flags.bits.PeakDetect == 0))
                            {
                                maxVal = localMax;      //
Save Value from Peak
                                maxFreq = localFreq;    //
Save Frequency from Peak

                                //* Set Peak Detected
                                flags.bits.PeakDetect = 1;
                                TP7_LAT = 1;
                            }

                        }
                    }
                }
            }
        }
        //* Increase Frequency by Step Size
        newFreq = (uint16_t)(NCO1INC + FREQ_STEP);
        OUTPUT_FREQ(newFreq);
    }
```

```
}
```

5. Run State Codes

```
#include "mcc_generated_files/mcc.h"
#include "nebulizer.h"


// Global RAM
adc_result_t batteryVoltage;
uint16_t maxTreatmentTimer;
uint8_t TimerValue;


void RUN_State(void)
{
    //* Read Battery Voltage
    batteryVoltage = ADC_GetConversion(channel_FVRBuffer1);


    if (SWITCH_PORT == 0)           //* Is Button Pressed?
    {
        //* Flag Button Press
        flags.bits.ButtonPress = 1;

        //* Set 20ms Timer
        setTimer(MS20);

        flags.bits.Debounce = 1;
    }
    if (flags.bits.Debounce)       //* Button Press Timer
Expired?
    {
        if (SWITCH_PORT)       //* Is Button Released
        {
            flags.bits.Debounce = 0;

            //* State = STOP
            state = STOP;
        }
    }


    if (batteryVoltage < LOW_BATTERY_P_HYST)           //* Is
Battery > 3.5v?
    {
        //* Red LED Off
        RED_LED_OFF();

        //* Green LED On
```

```
        GRN_LED_ON();

        PWM4_LoadDutyValue(FULLPOWER);           //* Adjust
Duty Cycle to Battery Voltage
    }
    else if(batteryVoltage > SHUTDOWN_P_HYST)            //*
Is Battery < 2.9v?
    {
        //* Set Indication to Red Flash
        flags.bits.FlashRed = 1;
        flags.bits.FlashGrn = 0;

        //* State = STOP
        state = STOP;
    }
    else
    {
        //* Red LED On
        RED_LED_ON();

        //* Green LED Off
        GRN_LED_OFF();

        PWM4_LoadDutyValue(REDUCEDPOWER);           //* Adjust
Duty Cycle to Battery Voltage
    }

    if(TMR0IF)
    {
        TMR0IF = 0;

        if(TimerValue)
        {
            TimerValue--;
        }
        if(TimerValue == 0)
        {
            TimerValue = 30;

            if(maxTreatmentTimer)
            {
                // TMR0 = 0;
                TMR0_Reload();

                maxTreatmentTimer--;

                if(maxTreatmentTimer == 0)      // Max
Treatment Timer Expired?
                {
                    maxTreatmentTimer = 120;

                    GRN_LED_OFF();
                    //* Set Indication to Green Flash
                    flags.bits.FlashRed = 0;
                    flags.bits.FlashGrn = 1;
```

```
                    //* State = STOP
                    state = STOP;
                }
            }
        }
    }
}
```

6. Stop State Codes

```
#include "mcc_generated_files/mcc.h"
#include "nebulizer.h"

void STOP_State(void)
{
    //* Disable VSUPPLY
    POWER_OFF;

    //* Disable BOOST
    BOOST_OFF;
    flags.bits.BoostReady = 0;

    //* Disable OUTPUT
    OUTPUT_OFF;

    //* Disable Comparator DACs
    DAC1_OFF();
    DAC2_OFF();

    //* Disable FVR
    FVR_OFF();

    //* State = INDICATE
    state = INDICATE;
}
```

7. Indicate State Codes

```
#include "mcc_generated_files/mcc.h"
#include "nebulizer.h"

uint8_t TimerValue = 30;
uint8_t FlashValue = 20;

void INDICATE_State(void)
{
    if(flags.bits.ButtonPress)    //* Is button pressed?
    {
        flags.bits.ButtonPress = 0;
        //* Set Indication to None
        flags.bits.FlashRed = 0;
        flags.bits.FlashGrn = 0;
        flags.bits.FlashToggle = 0;
```

```
        RED_LED_OFF();
        GRN_LED_OFF();

        //* State = SLEEP
        state = SLEEP;
    }


    if(TMR0IF)
    {
        TMR0IF = 0;

        if(TimerValue)
        {
            TimerValue--;
        }
        if(TimerValue == 0)
        {
            TimerValue = 30;

            if(FlashValue)
            {
                // TMR0 = 0;
                TMR0_Reload();

                if(flags.bits.FlashToggle)
                {
                    LED_RED_Toggle();
                    LED_GRN_Toggle();
                }
                else
                {
                    if(flags.bits.FlashRed)
                    {
                        LED_RED_Toggle();
                        GRN_LED_OFF();
                    }
                    else if(flags.bits.FlashGrn)
                    {
                        LED_GRN_Toggle();
                        RED_LED_OFF();
                    }
                    else
                    {
                        RED_LED_OFF();
                        GRN_LED_OFF();
                    }
                }

                FlashValue--;

                if(FlashValue == 0)
                {
                    FlashValue = 20;
```

```
                        // Set Indication to None
                        flags.bits.FlashRed = 0;
                        flags.bits.FlashGrn = 0;
                        flags.bits.FlashToggle = 0;

                        RED_LED_OFF();
                        GRN_LED_OFF();

                        //* State = SLEEP
                        state = SLEEP;
                    }
                }
            }
        }
}
```

8. Sleep State Codes

```
#include "mcc_generated_files/mcc.h"
#include "nebulizer.h"

void SLEEP_State(void)
{
    if(SWITCH_PORT)     //* Is Button Released?
    {
        //* Set Chip to Wake Up on Button Press
        IOCCF = 0;
        IOCCNbits.IOCCN0 = 1;
        INTCONbits.IOCIF = 0;
        INTCONbits.IOCIE = 1;

        //* Enter Sleep Mode
#ifdef __DEBUG
        while (INTCONbits.IOCIF == 0)
        { }
#else
        SLEEP();
#endif

        INTCONbits.IOCIE = 0;   // Disable Interrupt On Change
        __delay_ms(20);         // Debounce Switch

        if(SWITCH_PORT == 0)    //* Is Button Pressed?
        {
            while (SWITCH_PORT == 0)        // Wait for Button
Release
            {
                __delay_ms(20);     //* Delay 20ms
            }

            FVR_Initialize();
```

```
            ADC_Ref_VDD();          // Set ADC Reference to
VDD

            if (ADC_GetConversion(channel_FVRBuffer1) >
SHUTDOWN_P_HYST)      //* Is Battery < 2.9v?
            {
                //* Set Indication to Short Red Flash
                flags.bits.FlashRed = 1;
                flags.bits.FlashGrn = 0;

                //* State = INDICATE
                state = INDICATE;
            }
            else
            {
                //* State = INIT
                state = INIT;
            }
        }
    }

}
```