A PROJECT REPORT ON

GENERATION OF PULSE WIDTH MODULATION
USING MICROCONTROLLER AND GENERATE
SINUSOIDAL WAVE

In Partial Fulfillment of the Requirements for the Degree of Bachelor of
Science in Electrical and Electronic Engineering, February 2012

SUMMITED BY

NIZAM UDDIN (ID-081800070)
ABU NIZAM BHUIYAN (ID-083800049)
JAHANGIR TARAFAQDAR (ID-083800056)
MOFIZUR RAHMAN (ID-083800071)
NOOR-E-ALAM (ID-083800078)

SUPERVISED BY

Prof. Dr. Mirza Golam Rabbani
Chairperson (Dean In-Charge)
EEE Department
E & T Faculty
Eastern University, Bangladesh

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EASTERN UNIVERSITY, DHAKA
Declaration

I hereby declare that this thesis is based on the results I found in my thesis work. Contents of work found by other researcher are mentioned by reference. This thesis paper has never been previously submitted for any degree neither in whole nor in part.

__________________________  ____________________________
NIZAM UDDIN                  ABU NIZAM BHUIYAN
ID: 081800070                ID: 083800049

__________________________  ____________________________
JAHANGIR TARAFDAR            MOFIZUR RAHMAN
ID: 083800056                ID: 083800071

__________________________
NOOR-E-ALAM
ID: 083800078

Supervisor                    External

__________________________  ____________________________
Prof. Dr. Mirza Golam Rabban  S.M. Shakil Hassan
Chairperson (Dean In-Charge),  Lecturer
EEE Department               EEE Department
E & T Faculty                E & T Faculty
Eastern University, Bangladesh Eastern University, Bangladesh
Approval

The project report design and implementation of “Generation Of PWM Pulses Using Microcontroller For Better Harmonic Performance Of Instantaneous Power Supply” has submitted to the following respected members of the board of examiners of the faculty of engineering in partial fulfillment of the requirements for the degree of Bachelor of Science in Electrical & Electronics Engineering on August of 2011 by the following students and has accepted as satisfactory.

1. ABU NIZAM BHUIYAN ID: 083800059

2. JAHANGIR TARAFC DAR ID: 083800056

3. MOFIZUR RHAMAN ID: 083800071

4. NOOR-E-ALAM ID: 083800078

5. NIZAM UDDIN ID: 081800070

Supervisor

Prof. Dr. Mirza Golam Rabbani
Chairperson (Dean In-charge),
EEE Department,
E & T Faculty,
Eastern University, Bangladesh
Acknowledgement

Firstly, I would like to thank my supervisor Prof. Dr. Mirza Golam Rabbani for giving us the opportunity to work on this project under his supervision and also for his invaluable support and guidance throughout the period of pre-thesis and thesis semesters. Through his supervision, I have learned a lot.

Lastly, I would like to thank our Lab Technical Officers for their support and guidance in my project.

Authors
Abstract

Energy crisis are of special attention in today's World. The unending usage of non-renewable energy sources will bring an end to the limited resources in near future. In order to preserve the resources, several alternative renewable sources have been in use these days. The power generated from the renewable sources, like solar energy, produces is a DC power which can be stored in batteries. This DC power needs to be converted to AC power as most of the appliances used in our daily life are dependent on AC power. To overcome this obstacle, DC-AC Inverter took birth.

Inverters can be categorized into three groups: Square wave, modified sine wave and pure sine wave. Considering power wattage, efficiency and harmonic content, pure sine wave inverters has proved to have the best quality among the three types. The control circuit for pure sine wave Inverter produces sinusoidal pulse width modulation. There are two basic topologies to generate pulse width modulation Topology 1. Analog Control circuit and Topology 2: Microcontroller based control circuit.

In this thesis project, performance of both topologies used in inverters has been analyzed and a DC-AC pure sine wave Inverter using the Analog control circuit (Topology 1) has been implemented.
# Table of Content

<table>
<thead>
<tr>
<th>Sl#</th>
<th>CHAPTER 1</th>
<th>Page No</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Motivation</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>1.3</td>
<td>Types of Inverter</td>
<td>4</td>
</tr>
<tr>
<td>1.4</td>
<td>Topologies of Pure Sine Wave Inverters</td>
<td>7</td>
</tr>
<tr>
<td>1.5</td>
<td>Topology 1: Analog Control Circuit</td>
<td>7</td>
</tr>
<tr>
<td>1.6</td>
<td>Topology 2: Microcontroller Based Control Circuit</td>
<td>8</td>
</tr>
</tbody>
</table>

**CHAPTER 2**

| 2.1 | Design Method of pure sine wave inverter (Topology 1) | 10       |
| 2.2 | Block Diagram | 10       |
| 2.3 | Software simulation | 11       |
| 2.4 | Graphical Output of Software simulation | 12       |
| 2.5 | Sine Wave generator | 13       |
| 2.6 | Carrier Wave Generator | 14       |
| 2.7 | Pulse Width Modulation | 16       |
| 2.8 | H-Bridge | 17       |
| 2.9 | Filter | 19       |

**CHAPTER 3**

| 3.1 | Implementation of Design | 22       |
| 3.2 | Results | 24       |
| 3.3 | Difficulties | 24       |

**CHAPTER 4**

<p>| 4.1 | ATMega8, Pin Diagram of Microcontroller | 26       |</p>
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>Features</td>
<td>28</td>
</tr>
<tr>
<td>4.3</td>
<td>Overview</td>
<td>29</td>
</tr>
<tr>
<td>4.4</td>
<td>Disclaimer</td>
<td>32</td>
</tr>
<tr>
<td>4.5</td>
<td>Pin Description</td>
<td>32</td>
</tr>
<tr>
<td>4.6</td>
<td>Resisters</td>
<td>34</td>
</tr>
<tr>
<td>4.7</td>
<td>AVR AT Mega8 Memories</td>
<td>38</td>
</tr>
<tr>
<td>4.8</td>
<td>EEPROM Read/Write/Access</td>
<td>40</td>
</tr>
<tr>
<td>5.1</td>
<td>Microcontroller Based Inverter: (Topology 2)</td>
<td>44</td>
</tr>
<tr>
<td>5.2</td>
<td>General Description of Sinusoidal PWM</td>
<td>44</td>
</tr>
<tr>
<td>5.3</td>
<td>Basic Design Of Microcontroller Based Inverter</td>
<td>45</td>
</tr>
<tr>
<td>5.4</td>
<td>Methodology</td>
<td>45</td>
</tr>
<tr>
<td>5.5</td>
<td>Algorithm For Generating PWM</td>
<td>47</td>
</tr>
<tr>
<td>5.6</td>
<td>Necessary Software’s</td>
<td>48</td>
</tr>
<tr>
<td>5.7</td>
<td>Description of This Software’s</td>
<td>48</td>
</tr>
<tr>
<td>5.8</td>
<td>How to write the code in AVR studio</td>
<td>49</td>
</tr>
<tr>
<td>5.9</td>
<td>Basic Code</td>
<td>51</td>
</tr>
<tr>
<td>5.10</td>
<td>Coding Of Sinusoid PWM</td>
<td>52</td>
</tr>
<tr>
<td>5.11</td>
<td>Implementation of Sinusoidal Wave</td>
<td>54</td>
</tr>
<tr>
<td>5.12</td>
<td>Results</td>
<td>54</td>
</tr>
<tr>
<td>5.13</td>
<td>Block Diagram Of Microcontroller Based Inverter</td>
<td>55</td>
</tr>
<tr>
<td>5.14</td>
<td>Future Works</td>
<td>56</td>
</tr>
<tr>
<td>5.15</td>
<td>Conclusion</td>
<td>56</td>
</tr>
</tbody>
</table>
CHAPTER – 1

- Motivation
- Introduction
- Types of Inverter
- Topologies of Pure Sine Wave Inverters
- Topology 1: Analog Control Circuit
- Topology 2: Microcontroller Based Control Circuit
1.1 Motivation

Bangladesh is suffering from a shortage of electricity and to overcome this crisis, alternative sources of energy are expanding, especially in rural areas of Bangladesh. The Solar Home System is one of the alternative sources that can mitigate the demand for electricity, especially in rural areas where electricity has not reached yet.

Solar energy is a renewable energy without causing pollution to the environment. The maximum electricity that a solar panel can produce is 130 Watt (130 wup). By this panel, 11 CFL (compact fluorescent lamp) of 6 watt power and a 17-20 inches back and white TV can run. Fan conducted on DC current can also be run by this solar energy. BRAC Solar Home System, Grameen Sakti, and few other companies are working to provide solar energy to the villages in Bangladesh. Their main objective is to provide electricity all over Bangladesh.

After some research, we found that most of the companies, including BRAC Solar Home System, are dependent on DC appliances. Due to lack of proper inverters, companies provide usages of DC appliances only and not AC appliances. That is because the existing inverters produce modified sine wave (square waves) which causes a power loss and harms the AC appliances. It is learnt that the amount of investment in this energy sector in the entire area per year is more than 2500 crore, 60% of this invested in solar panel, total of which requires being imported from outside. 25% is invested in battery and the rest 15% in small mechanical parts. The companies are interested to extend the service to the city dwellers.

In the near future, the demand of pure sine wave inverter will be sky-rocketing, since most of the appliances around us are dependent on AC power. Thus, this has motivated us to create a pure sine wave inverter which can be implemented in Solar Home Systems, at an affordable cost so the rural people can be benefited besides others. Our goal is to produce pure sine wave, and not modified. If we can successfully implement the analogue circuit, then we can digitize our pure sine wave inverter circuitry using micro-controller applications.
1.2 INTRODUCTION

The output of battery source is a 12 volt DC and the required AC volt for AC appliances is a 220V AC (50 Hz). The function of inverter is to convert 12 DC to 220V AC which should have pure sine wave oscillation at 50 hertz like the ordinary household electrical outlet. The method that we are applying in our experiment is, converting the low voltage DC power to AC, and then using a transformer to boost the voltage to 220V AC.

In today's market, there are two different types of inverters, modified sine wave and pure sine wave inverter. The modified Sine wave is similar to a square wave which is less efficient in power consumption. It produces high number of harmonies which affects the devices, hence, reducing its life time. Whereas, a pure sin wave inverter reduces the harmonics, to minimum, thus increasing the efficiency of power consumption and life time of AC appliances. It also reduces the audible and electrical noise in audible equipment, TV's, Fluorescent lights and allows inductive load, like fan to run faster and quieter. The modified basic differences between the three waves are shown in the figure: 1, below. It can be noted that the square wave has the minimum amplitude, that is, maximum distortion. The modified sin wave is quite similar to square, but it rests at zero for moment then rises or falls, it's less distorted then the square. Whereas, the pure sin wave shown, have zero distortion (maximum amplitude) compared to the rest two with pure oscillation of sine wave.
1.3 Types of Inverter:

DC to AC Inversion

Square Wave Inverters

DC to AC conversion is most commonly done through use of MOSFET inverter circuits, which can switch the voltage across the load, providing a digital approximation of the desired AC signal. The simplest variant of this inversion is the production of a square wave approximation of a sine wave (Figure 1). For a square wave, the load voltage must be switched merely from high to low, without the need for an intermediate step (i.e. 0V). In order to deliver the same power as the sine wave to be approximated, the amplitude of the square wave must be the sine wave's RMS value. This way, the average voltages, and therefore the power delivered, will be the same for the two waveforms. Square wave inverters are very rarely used in practice, as many devices which utilize timing circuits that rely on something close to the sine wave from the power company cannot operate with such a rough approximation. In addition, a square wave has relatively large 3rd and 5th harmonic components (figure 2), which burn power and severely cut down on the efficiency of devices using such inverters as a power source.

![Figure 1: Square Wave](image-url)
Modified Sine Wave Inverters

A very common upgrade to the square wave inverter is the modified sine wave inverter. In the modified sine wave inverter, there are three voltage levels in the output waveform, high, low, and zero (figure 3), with a dead zone between the high and low pulses. The modified sine wave is a closer approximation of a true sine wave than is a square wave, and can be used by most household electrical devices. As such, it is extremely common to see this type of inversion in commercial quality inverters. Despite being much more viable than a simple square wave, the modified sine wave has some serious drawbacks. Like the square wave, modified sine waves have a large amount of power efficiency loss due to significant harmonic frequencies (figure 4), and devices that rely on the input power waveform for a clock timer will often not work properly. Despite the inherent drawbacks, many devices can work while powered by a modified sine source. This makes it an affordable design option for such implementations as household uninterruptible power supplies.
Pure Sine Wave Inverters

The best power source for most applications is a pure 60Hz sine wave, identical to the 120Vrms source available from any US power company. All low power household plug-in devices are designed to work with this source (high power devices such as cooking ovens use a 240V source) and, as such, will be most likely to work properly and most efficiently on such a source. A true sine wave source is produced most easily for high power applications through rotating electrical machinery such as naval gas-turbine generators, house-hold diesel or gasoline backup generators, or the various generators employed by power companies that employ a shaft torque to create an AC current. These sources provide a relatively clean, pure sine wave (lacking significant harmonics and high frequency noise) thanks to their analog rotational make-up. Such rotating machinery can be inappropriate for low-power backup supply usage due to their high cost, large size and required maintenance. As such, a smaller, digital pure sine wave inverter can be extremely useful.
1.4 Topologies of Pure Sine Wave Inverter:

Pulse width modulation is widely used as a source of powering alternating current (AC) devices with available direct current (DC) source. Variation of duty cycle of the PWM signal to provide dc voltages across the load in a specific pattern will appear to the load as ac signal. The pattern at which the duty cycle of a PWM Signal varies can be implemented using simple analog components or a digital microcontroller. There are two basic topologies in generate sinusoidal PWM that controls that output of the inverter. The two topologies are,

1.5 TOPOLOGY 1: Analog Control Circuit

In this type of control circuit, the SPWM is generated by comparing a sine wave with a high carrier frequency (Triangular wave or saw tooth wave) with sine wave as the reference voltage. Figure (6) shows how PWM A produced by comparing sine wave with triangular wave. This type of topology, analog components is used to generate the sine wave and carrier wave and compare them with analog comparator.
1.6 TOPOLOGY 2: Microcontroller Based Control Circuit

In this type of topology, the SPWM is directly achieved using microcontroller that will control the final sine wave at the final output. Using this type of topology reduces the cost and the size of the control circuit. The low cost micro controller like ATmega8 or Pic18F4431 has built in PWM modules which require some command to generate the necessary PWM wave form.
CHAPTER 2

- Design Method of pure sine wave inverter (Topology 1)
- Block Diagram
- Software simulation
- Graphical Output of Software simulation
- Sine Wave generator
- Carrier Wave Generator
- Pulse Width Modulation
- H-Bridge
- Filter
2.1 Design Method of pure sine wave Inverter (Topology 1)

The Implementation of pure sine wave Inverter can be complex when thought of as a whole but when broken into smaller projects and implementing them individually, it becomes easier. the following sections describes how the Individual blocks like Oscillator section, carrier wave generator and switching circuit was Implemented and how the blocks were joined together.

2.2 Block Diagram:

Our project consists of analog circuitry (resistors, capacitor, diode, variable resistors) as well as discrete components (Integrated circuits-LM348, TU84, IAC331p, 144OSFIET driver (111-21101), MOSFETs and step up transformer. Thai is all required to construct a sin wave to generate 220V AC sine wave across a load. The block diagram (Figure 1) illustrates the various parts or blocks of the project. The three basic blocks to control the circuit are the Six volt reference, sine wave generator and triangle wave generator. When these blocks are implemented with comparator, and other small circuitry, they control the pulse width modulated (PWIM) signals that are fed into two MOSFET drivers, The comparator circuit also produces square waves that are fed into other MOSFET drivers which determine the polarity of the final output sin wave. 1% PWIM signals fed into the MOSFET Driver performs the level translation to drive four N-Channel h4OSFETs in an H-Bridge configuration. From here, the signals from the N-Channel MOSFETs are passed through a low pass filter so that the output is a pure sin wave of 12V P-P with 50 hertz frequency. Then the signal will be boosted up to 220 V AC using a step-up transformer.
2.3 SOFTWARE Simulations

In order to understand the circuit we had to design the circuit in Pspice software and note the outputs some specific nodes. The IC used in the hardware implementation of comparator circuit was MC3302 which has 4, comparator OP-AMPS inside it. But due to unavailability of MC3302 in Pspice library, we used LM 139 which is the same series of 1140302. The control circuit (sine wave generator, carrier triangular wave generator, and 6V reference) was artificially created using VSIN, VPULSE and VDC from the library. Then the signals were fed into the comparator circuit.
2.4 GRAPHICAL OUTPUTS OF SOFTWARE SIMULATION

The outputs of the four comparator OP-AMPS were carefully noted by varying the transient time and other parameters. Finally the expected results were crystal clear, that our system will work, that is, we can generate PWM and Square wave if we go for hardware implementation. Figure – 9.1, shows the bath the square wave and inverted square wave of 12 V p-p and frequency is dependent on the frequency of the Sine wave generator. In this case, the frequency was 6QHZ. And Figure 9.2: shows the PWM and inverted PWM (12V p-p). Bath of the outputs are shown in the same axis in order to understand the inversion of respective waves.

Figure - 9:1: Square wave fed to left MOSFET driver which controls the polarity of output sine wave.
Figure - 9.2: PWM fed in to right side MOSFET driver which controls the amplitude of output sine wave.

2.5 Sine Wave generator:
For sine wave generator block, we have used Bubba Oscillator. The bubba Oscillator is a circuit that provides a filtered sin wave of any frequency based on the configuration of resistors and capacitors. The Bubba oscillator is a phase-shift circuit which requires a 45 degree phase-shift in order to function. The four OP-AMP when placed in series, produces a total 180 degree phase-shift. The biggest advantage of Bubba Oscillator is that frequency stability holds while still giving a low distortion output. The RC filter used after each OP-Amp provides clear and stable signals. The four identical IRICC filters phase-shifts the signal to 45 degree each. This causes a 180 degree phase shift which is then returned to a zero phase shift with the inverting amplifier across the first OP-AMPS.
Figure 10: Circuit diagram of the Bubba Oscillator

OUTPUTS OF THE BUBBA OSCILLATOR

Figure 11: Output of the Bubba Oscillator taken from the Digital Oscilloscope.

2.6 CARRIER WAVE GENERATOR

Carrier waves can be either saw tooth or triangular signals; in our experiment, a triangular wave will is used. This wave is set to 50 KHz to determined optimal power
loss. The generation of the triangular carrier wave will be done with analog components. The circuit for the construction of the triangle wave generator consists of a square wave generator and integrator, as shown in Figure 12. The above circuit will oscillate at a frequency of $1/4f_{ttC}$, and the amplitude can be controlled by the ratio of $R1$ and $R2$.

![Triangle Wave Generator](image1.png)

**Figure 12 a: Triangle Wave Generator**

**OUTPUT OF THE TRIANGLE WAVW GENERATOR:**

![Output of Triangle Wave Generator](image2.png)

**Figure 12 b: Output of the Triangle wave generator taken from the Digital Oscilloscope.**
2.7 Pulse Width Modulation:

In this project a Tri level PWM is produced by comparing a modified triangular wave with the reference sine wave.

Figure 13: Sinusoid Pulse Width Modulation

In figure () the top picture shows the input reference wave form and the generated PWM signal. The bottom picture shows the signals which are passed into comparator to achieve the PWM wave form. The triangular wave must be modified such that it switch’s between a mid-to-high triangular wave mid-to-low. Figure () shows the modified triangular wave that was achieved in the experiment.

Figure 14: Modified Triangular wave
Now with this modified Triangular wave, a sine is compared using comparators and the PWM achieved in the experiment, is shown in figure ():

![Figure 15: Output of comparator](image)

2.8 H-Bridge:
An H-bridge is an electronic circuit which enables a voltage to be applied across a load in either direction. The term "H-bridge" is derived from the typical graphical representation of such a circuit. An H-bridge is built with four switches (solid-state or mechanical). When the switches S1 and S4 (according to the first figure) are closed (and S2 and S3 are open) a positive voltage will be applied across the motor. By opening S1 and S4 switches and closing S2 and S3 switches, this voltage is reversed, allowing Reverse operation of the motor.

Using the nomenclature above, the switches S1 and S2 should never be closed at the same time, as this would cause a short circuit on the input voltage source. The same applies to the switches S3 and S4. This condition is known as shoot-through.
Generating a sine wave centered on zero volts require a positive, and negative voltage across the load, for the positive and negative parts of the wave, respectively. T16s can be implemented, from a single source through the use of MOSFET switches arranged in an FAB44ge configuration. To minimize power loss and utilize higher switching speeds, N-channel MOSFETS were chosen as switches in the bridge. To drive the HIGH side of the H-Bridge, MOSFET Drivers were used. Figure (1) shows the connections of the MOSFET driver to the H-Bridge.

The inputs to the MOSFET Drivers are shown in Figure () and figure ()
2:9 Filters

A low-pass fitter is a fitter that passes low-frequency signals but attenuates signals with frequencies higher than the cutoff frequency. The actual amount of attenuation for each frequency varies from filter to filter. It is sometimes called a high-cut filter or treble outfitter when used in audio applications. A low-pass fitter is the opposite of a high-pass fitter, and a band-pass fitter is a combination of a low-pass and a high-pass.

In this experiment, a LC low pass fitter was used which extracts the high frequency carrier wave from the original signal.
Figure 20: LC Low pass filter

The signals entering the LC- filter is shown in Figure below,

Figure 21: Output of MOSFET switching circuit
CHAPTER – 3

- Implementation of Design
- Results
- Difficulties
3.1 Implementation of Design

Design of H-Bridge Circuit:

Figure 22: H-Bridge Circuit

Design of Microcontroller Circuit:

Figure 23: Microcontroller Circuit
Design of Filter Circuit:

![Filter Circuit](image)

Figure 24: Filter Circuit

Design of Full Circuit:

![Full Circuit](image)

Figure 25: Full Circuit
After successfully completing the task of individual sections on different circuit board joked together accordingly. The first board H-Bridge is get supply and pass 12 volt to the Microcontroller board then it convert 12 volt to 5 volt to supply microcontroller and it generate pulse and send to H-Bridge again then through the Filter circuit it generate our required PWM signal.

3.2 Results

The final result was a sine wave of 60 HZ with an amplitude of 10-12V p-p depending on the frequency of the desire output with little distortion but pure indeed for most of the time.

3.3 Difficulties

The most difficult part was to design the LC filter since the type Inductor used is not available. So whenever the circuit was turned with wrong filter load, huge current flowed through the circuit causing damages to ICs.
CHAPTER 4

- AT Mega8, Pin Diagram of Microcontroller
- Features
- Overview
- Disclaimer
- Pin Description
- Resisters
- AVR AT Mega8 Memories
- EEPROM Read/Write/Access
4.1 ATmega8 Pin Diagram of Microcontroller:

Figure 26: Microcontroller Circuit Pin Diagram
Figure 27: Microcontroller Circuit MLF Top View Pin Diagram

Figure 27: Microcontroller Circuit TQFP Top View Pin Diagram
4.2 Features

• High-performance, Low-power AVR® 8-bit Microcontroller

• Advanced RISC Architecture
  – 130 Powerful Instructions
  – Most Single-clock Cycle Execution
  – 32 x 8 General Purpose Working Registers
  – Fully Static Operation
  – Up to 16 MIPS Throughput at 16 MHz
  – On-chip 2-cycle Multiplier

• High Endurance Non-volatile Memory segments
  – 8K Bytes of In-System Self-programmable Flash program memory
  – 512 Bytes EEPROM
  – 1K Byte Internal SRAM
  – Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  – Data retention: 20 years at 85°C/100 years at 25°C
  – Optional Boot Code Section with Independent Lock Bits
  – In-System Programming by On-chip Boot Program
  – True Read-While-Write Operation
  – Programming Lock for Software Security

• Peripheral Features
  – Two 8-bit Timer/Counters with Separate Prescaler, one Compare Mode
  – One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  – Real Time Counter with Separate Oscillator
  – Three PWM Channels
  – 8-channel ADC in TQFP and QFN/MLF package

• Eight Channels 10-bit Accuracy
  – 6-channel ADC in PDIP package

• Six Channels 10-bit Accuracy
  – Byte-oriented Two-wire Serial Interface
  – Programmable Serial USART
- Master/Slave SPI Serial Interface
- Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator

**Special Microcontroller Features**
- Power-on Reset and Programmable Brown-out Detection
- Internal Calibrated RC Oscillator
- External and Internal Interrupt Sources
- Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby

**I/O and Packages**
- 23 Programmable I/O Lines
- 28-lead PDIP, 32-lead TQFP, and 32-pad QFN/MLF

**Operating Voltages**
- 2.7 - 5.5V (ATmega8L)
- 4.5 - 5.5V (ATmega8)

**Speed Grades**
- 0 - 8 MHz (ATmega8L)
- 0 - 16 MHz (ATmega8)

**Power Consumption at 4 Mhz, 3V, 25° C**
- Active: 3.6 mA
- Idle Mode: 1.0 mA
- Power-down Mode: 0.5 μA

### 4.3 Overview

The ATmega8 is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega8 achieves throughputs approaching 1 MIPS per MHz, allowing the system designer to optimize power consumption versus processing speed.

**Block Diagram of Microcontroller**
Figure 28: Microcontroller Block Diagram
The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers are. The ATmega8 provides the following features: 8K bytes of In-System Programmable Flash with Read-While-Write capabilities, 512 bytes of EEPROM, 1K byte of SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte oriented Two wire Serial Interface, a 6-channel ADC (eight channels in TQFP and QFN/MLF packages) with 10-bit accuracy, a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and five software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next Interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. The device is manufactured using Atmel’s high density non-volatile memory technology. The Flash Program memory can be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile memory programmer, or by an On-chip boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash Section will continue to run while the Application Flash Section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega8 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.
The ATmega8 AVR is supported with a full suite of program and system development tools, including C compilers, macro assemblers, program debugger/simulators, In-Circuit Emulators, and evaluation kits.

4.4 Disclaimer
Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

4.5 Pin Descriptions
VCC Digital supply voltage.
GND Ground.
Port B (PB7..PB0)
XTAL1/XTAL2/TOSC1/TOSC2
Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running. Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit. Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier.
If the Internal Calibrated RC Oscillator is used as chip clock source, PB7..6 is used as TOSC2..1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set. The various special features of Port B are elaborated in “Alternate Functions of Port B”

Port C (PC5..PC0)
Port C is an 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.
PC6/RESET
If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C. If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. The minimum pulse length is given. Shorter pulses are not guaranteed to generate a Reset.

Port D (PD7..PD0)
Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

RESET
Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset.

AVCC
AVCC is the supply voltage pin for the A/D Converter, Port C (3.0), and ADC (7..6). It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter. Note that Port C (5..4) use digital supply voltage, VCC.

AREF
AREF is the analog reference pin for the A/D Converter.

ADC7..6 (TQFP and QFN/MLF Package Only)
In the TQFP and QFN/MLF package, ADC7..6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.
**About Code Examples**

This datasheet contains simple code examples that briefly show how to use various parts of the device. These code examples assume that the part specific header file is included before compilation. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C compiler documentation for more details.

**4.6 Registers**

**Status Register**

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code. The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

The AVR Status Register – SREG – is defined as:

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>T</td>
<td>H</td>
<td>S</td>
<td>V</td>
<td>N</td>
<td>Z</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>Initial Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bit 7 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the Instruction Set Reference.
**Bit 6 – T: Bit Copy Storage**
The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit. A bit from a register in the Register File can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

**Bit 5 – H: Half Carry Flag**
The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half Carry is useful in BCD arithmetic. See the “Instruction Set Description” for detailed information.

**Bit 4 – S: Sign Bit, S = N \( \lor \) V**
The S-bit is always an exclusive or between the Negative Flag N and the Two’s Complement Overflow Flag V. See the “Instruction Set Description” for detailed information.

**Bit 3 – V: Two’s Complement Overflow Flag**
The Two’s Complement Overflow Flag V supports two’s complement arithmetics. See the “Instruction Set Description” for detailed information.

**Bit 2 – N: Negative Flag**
The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

**Bit 1 – Z: Zero Flag**
The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

**Bit 0 – C: Carry Flag**
The Carry Flag C indicates a Carry in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.
**General Purpose Register File**

The Register File is optimized for the AVR Enhanced RISC instruction set. In order to achieve the required performance and flexibility, the following input/output schemes are supported by the Register File:

- One 8-bit output operand and one 8-bit result input.
- Two 8-bit output operands and one 8-bit result input.
- Two 8-bit output operands and one 16-bit result input.
- One 16-bit output operand and one 16-bit result input.

Figure 29 shows the structure of the 32 general purpose working registers in the CPU.

![Figure 29: AVR CPU General Purpose Working Registers](image)

Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions. As shown in Figure 3, each register is also assigned a Data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y-, and Z-pointer Registers can be set to index any register in the file.
### The EEPROM Address Register – EEARH and EEARL

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EEARH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EEARL</td>
</tr>
<tr>
<td>Read/Write</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R/W</td>
</tr>
<tr>
<td>Initial Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

The EEPROM Address Registers – EEARH and EEARL – specify the EEPROM address in the 512 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 511. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

### The EEPROM Data Register – EEDR

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EEDR</td>
</tr>
<tr>
<td>Read/Write</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>Initial Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

For the EEPROM write operation, the EEDR Register contains the data to be written to the EEPROM in the address given by the EEAR Register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

### The EEPROM Control Register – EECR

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EECR</td>
</tr>
<tr>
<td>Read/Write</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>Initial Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>

These bits are reserved bits in the ATmega8 and will always read as zero.
4.7 AVR ATmega8 Memories

This section describes the different memories in the ATmega8. The AVR architecture has two main memory spaces, the Data memory and the Program Memory space. In addition, the ATmega8 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

In-System Reprogrammable Flash Program Memory

The ATmega8 contains 8K bytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16- or 32-bits wide, the Flash is organized as 4K x 16 bits. For software security, the Flash Program memory space is divided into two sections, Boot Program section and Application Program section. The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega8 Program Counter (PC) is 12 bits wide, thus addressing the 4K Program memory locations. The operation of Boot Program section and associated Boot Lock Bits for software protection are described in detail in “Boot Loader Support – Read-While-Write Self-Programming”

Figure 30: Programme Memory Map
SRAM Data Memory

Figure 31 shows how the ATmega8 SRAM Memory is organized. The lower 1120 Data memory locations address the Register File, the I/O Memory, and the internal data SRAM. The first 96 locations address the Register File and I/O Memory, and the next 1024 locations address the internal data SRAM. The five different addressing modes for the Data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement, and Indirect with Post-increment. In the Register File, registers R26 to R31 feature the indirect addressing pointer registers. The direct addressing reaches the entire data space. The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y- or Z-register. When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y and Z are decremented or incremented. The 32 general purpose working registers, 64 I/O Registers, and the 1024 bytes of internal data SRAM in the ATmega8 are all accessible through all these addressing modes.

![Figure 31: Data Memory Map](image-url)
Data Memory Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clk CPU cycles as described in Figure 32.

Figure 32: On-chip SRAM Data Access Cycle

EEPROM Data Memory

The ATmega8 contains 512 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described below, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

4.8 EEPROM Read/Write/Access

The EEPROM Access Registers are accessible in the I/O space. The write access time for the EEPROM is given in Table 1 on page 21. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, VCC is likely to rise or fall slowly on Power-up/down. This
causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this. When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

**Bit 3 – EERIE: EEPROM Ready Interrupt Enable**
Writing EERIE to one enables the EEPROM Ready Interrupt if the I bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready interrupt generates a constant interrupt when EEWE is cleared.

**Bit 2 – EEMWE: EEPROM Master Write Enable**
The EEMWE bit determines whether setting EEWE to one causes the EEPROM to be written. When EEMWE is set, setting EEWE within four clock cycles will write data to the EEPROM at the selected address. If EEMWE is zero, setting EEWE will have no effect. When EEMWE has been written to one by software, hardware clears the bit to zero after four clock cycles. See the description of the EEWE bit for an EEPROM write procedure.

**Bit 1 – EEWE: EEPROM Write Enable**
The EEPROM Write Enable Signal EEWE is the write strobe to the EEPROM. When address and data are correctly set up, the EEWE bit must be written to one to write the value into the EEPROM. The EEMWE bit must be written to one before a logical one is written to EEWE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 3 and 4 is not essential):

1. Wait until EEWE becomes zero.
2. Wait until SPMEN in SPMCR becomes zero.
3. Write new EEPROM address to EEAR (optional).
4. Write new EEPROM data to EEDR (optional).
5. Write a logical one to the EEMWE bit while writing a zero to EEWE in EECR.
6. Within four clock cycles after setting EEMWE, write a logical one to EEWE.
The EEPROM can not be programmed during a CPU write to the Flash memory. The software must check that the Flash programming is completed before initiating a new EEPROM write.

Step 2 is only relevant if the software contains a boot loader allowing the CPU to program the Flash. If the Flash is never being updated by the CPU, step 2 can be omitted.

Caution: An interrupt between step 5 and step 6 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR or EEDR Register will be modified, causing the interrupted EEPROM access to fail. It is recommended to have the Global Interrupt Flag cleared during all the steps to avoid these problems.

When the write access time has elapsed, the EEWE bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEWE has been set, the CPU is halted for two cycles before the next instruction is executed.

**Bit 0 – EERE: EEPROM Read Enable**

The EEPROM Read Enable Signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR Register, the EERE bit must be written to a logic one to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed. The user should poll the EEWE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR Register. The calibrated Oscillator is used to time the EEPROM accesses.
CHAPTER 5

- Microcontroller Based Inverter: (Topology 2)
- General Description of Sinusoidal PWM
- Basic Design Of Microcontroller Based Inverter
- Methodology
- Algorithm For Generating SPWM
- Necessary Software’s
- Description of This Software’s
- How to write the code in AVR studio
- Basic Code
- Coding Of Sinusoid PWM
- Implementation of Sinusoidal PWM in Microcontroller
- Results
- Block Diagram Of Microcontroller Based Inverter
- Future Works
- Conclusion
5.1 Microcontroller Based Inverter: (Topology 2)

The control circuit in a pure sine wave inverter is designed using Microcontroller. The advantages of this inverter are the use of a low cost microcontroller that has built in PWM modules. In this experiment, Atmega8 was used that was able to store required commands to generate the necessary PWM waveforms. In Atmega8, PORT D has four output that produces PWM. In this experiment, only the control part was discussed and implemented and not the entire inverter.

5.2 General Description of Sinusoidal PWM

The Method used for creating the pure sine wave in Microcontroller is done through manipulation of mathematical representation of the original sine wave. It is done by dividing half of the sine wave into "m" number of segments (even number), where area under the first quarter of the sine wave (0- \( \pi/2 \)) resembles series of the form \([2n-1]\) where \( n=1,2,\ldots,m/2 \) while areas of the next quarter from \( \pi/2-\pi \) will resemble series of the form \([2n-1]\) where \( n=m/2,\ldots,1 \)

![Segmentation of sine wave](image-url)

Figure 33: Segmentation of sine wave
5.3 Basic Design of Microcontroller Based Inverter:

In the Microcontroller based inverter, the control circuit has to produced two types of SPWM. SPMW remains on for half of the cycle of the sine wave and for rest half of the cycle off, and the other SPWM is the vise versa. These two signals are faded to two sides of the MOSFETS in the H-bridge, switching circuit which changes the polarity at the output. The basic design of the microcontroller based inverter is shown in figure (4).

![Diagram of Basic Design of Microcontroller Based Inverter](image)

**Figure 34: Output of filter circuit**

5.4 Methodology:

A simple method has been applied to the two types of SPWM required in microcontroller based inverter. For a 50 Hz sine wave, the total time period is 0.02s. So one of the SPWM should represent half of the sin for the first (0.00-0.01) sec and the other SPWM should produce SPWM of half of the sin for rest (0.01-0.02) sec. The clock pulses produced in microcontroller can be controlled. Figure (35) shows the PWM of the first half cycle of a sine wave, And Figure (36) shows the two types of SPWM that has to be generate to control the microcontroller based inverter. The final desired output is shown in figure (37) after the H-Bridge switching circuit and Filter Circuit.
Figure 35: Half sine segmentation

Figure 36: Two types of Sinusoid PWM

Figure 37: Output of filter circuit
5.5 Algorithms for Generating PWM:

START

Initialize Variables

Initialize Port, Initialize PCPWM and set all interrupts

Initialize Sine Look up Table

Main Function

PWM Output

Reset

END

Figure 38: Algorithm
The Flow Chart shown in the above figure ( ) is the algorithm for a single phase sinusoidal PWM signal. In this flow chart "initialize variables" means initialize the user defined memory cell, "initialize Ports" means definition the output ports. Then comes the Initialize Sine Look Up table which stores sampling value sine wave. The these values will be given to the main function and thus PMW output is achieved and then it checks the logic if the loop has finished or not, If yes, it goes to the sine look up table and does the same functions, and or else END.

5.6 Necessary Software

1. WIN_AVR
2. AVR_Studio
3. AVR_lb
4. Pony Prog for AVR
5. Proteas
6. Tunup_Utilities_7.0.0799_PL

Firstly, we have to install Win_AVR software. Win_AVR is a compiler. We have installed above software according to sequence.

5.7 Description of This Software’s

1. _Win_AVR :
   _Win_AVR has two versions.
   (i) WinAVR-20060421-install.exe
   (ii) WinAVR-20071221-install.exe
   We have preferred WinAVR-20060421-install.exe. Because of WinAVR-20071221-install.exe is a latest version. It will occur at some problems. Then we have installed AVR_Studio.

2. _AVR_Studio :
   _AVR_Studio has two versions.
   (i) AVR_Studio_VER_4.12
   (ii) AVR_Studio_VER_4.13
   We have installed _AVR_Studio_VER_4.12
   We have written code in program-in-c.

3. _AVR_lb :
   After that, we have installed _AVR_lb in microcontroller
4  _Pony Prog for AVR:
Pony prog is an in system program (ISP) programmer. We have made program and downloaded hex file by pony prog in microcontroller.

5.  _Proteas:
_Proteas is very important software to see the signal, which includes with the external hardware of microcontroller.

6.  _Tuneup_Utilities_7.0.0799_PL:
_Tunup_Utilities_7.0.0799_PL Software used for cleaned the Registry. When we will install those softwares it may face some problems. If we uninstall AVR_Studio_VER_4.13 after installing AVR_Studio_VER_4.13. Moreover, after that, if we want to install again AVR_Studio_VER_4.13, then faces some problems. Actually, those four (_WIN_AVR, _AVR_Studio, _AVR_lb, _Pony Prog for AVR) softwares are used for learning AVR. On the other hand, the _Proteas and _Tunup_Utilities_7.0.0799_PL softwares are using for more facilities.

5.8  _How to write the code in AVR studio_

We drawn a block diagram of overall project
After some interval, the PORT C has executed high and low. After that, we have simulated the code then saved, compiled and run it at a time.
5.9 Basic Code

Void main (void)
{
    While (1)
    {
    }
}

After written the code, Save and compile at a time. Thus, we made the hex file.

#include <avr/io.h>

Void main (void)
{
    DDRC=0xFF;
    Unsigned int i;
    for (;;)
    {
        for (i=0;i<65535;i++)
        {
        }
    }
    PORTC=0xFF;
    for ( i=0;i<65535;i++)
    {
    }
    PORTC=0x00;
}

5.10 Coding Of Sinusoid PWM:

```c
#include <mega8.h>
#include <delay.h>

unsigned int s=0;

void main(void)
{
    PORTB=0x00;
    DDRB=0x06;

    TCCR1A=0xB1;
    TCCR1B=0x01;
    TCNT1H=0x00;
    TCNT1L=0x00;
    ICR1H=0x00;
    ICR1L=0x00;
    OCR1AH=0x00;
    OCR1AL=0x00;
    OCR1BH=0x00;
    OCR1BL=0x00;

    while (1)
    {
        for(s=45;s<240;s=s+3)
        {
            OCR1AL=s;
            OCR1BL=0xff;
            delay_us(63);
        }
        for(s=220;s>45;s=s-3)
        {
            OCR1AL=s;
            OCR1BL=0xff;
        }
    }
}
```
delay_us(63);

for(s=45; s<220; s=s+3)
{
    OCR1AL=0x00;
    OCR1BL=~s;
    delay_us(63);
}

for(s=220; s>45; s=s-3)
{
    OCR1AL=0x00;
    OCR1BL=~s;
    delay_us(63);
}

};
5.11 Implementation of Sinusoidal Wave:

The program was downloaded into the ATmega8 microcontroller using AVR studio 4. Then it was operated using 5V DC power supply at pin 7 and pin 8 was grounded. The outputs were taken from Pin 15 and pin 16 which were connected to H-Bridge input and the outputs were simultaneously taken using both channels from filter ckt.

5.12 Results:

The output of the Atmieg8 microcontroller was observed using oscilloscope. Figure () shows the output of PIN 18 in channel 2, and Output of PIN 19 in Channel. 1. Its observed that both the channel shows a half cycle sinusoidal wave with a phase shit of π that is when channel 2 is on channel 1 is off and vice versa.
5.13 Block Diagram of Microcontroller Based Inverter:

The figure ( ) shows the block diagram of a Microcontroller based pure sine wave inverter. Clock pulses are generated using crystal oscillator and is given to the Microcontroller. Battery start reset button tell the microcontroller when to turn ON the inverter and when off. Two types of SPWM is produced shown in Figure ( ) which are fed into switching MOSFETs which switches the polarity to the transformer. And then a low pass filter is used to get a pure sine wave at the output.

Figure 42: Block Diagram of Microcontroller based inverter
5.14 Future Works

There are still many tasks left to implement the microcontroller-based inverter. The two Sinusoidal PWM signals shown in Figure will be fed to MOSFET switching and observe if the MOFTEF can switch its polarity so fast. If the bipolar PWM is achieved, then an LC filter will be designed to eliminate the higher carrier in SPWM to get a pure sine wave. Then both types of Inverters using difficult topology can be compared.

5.15 Conclusion

The basic goal of this thesis has been achieved, which is the conversion of DC voltage to 60 Hz AC voltage. Both Topologies, that is analog control circuit and microcontroller-based control circuit, were implemented and studied and Ohms the microcontroller-based inverter will be implemented in future.