TEMPERATURE MEASUREMENT FOR SPACE APPLICATIONS USING CAPACITIVE SENSORS

Dissertation submitted in partial fulfillment of the award of the degree of

BACHELOR OF ENGINEERING

In

ELECTRONICS AND COMMUNICATION

Submitted by
AISHWARYA R.C.
AKHILA S RAO
AMITESH BHAGWAN

Under the guidance of

Mr. K.V MURALIDHAR
Assistant professor
Dept. of E & C
BIT, Bangalore

DEPARTMENT OF ELECTRONICS AND COMMUNICATION
Visveswaraiah Technological University, Belgaum

2008-2009

TEMPERATURE MEASUREMENT FOR SPACE APPLICATIONS USING CAPACITIVE SENSORS

Dissertation submitted in partial fulfillment of the award of the degree of

BACHELOR OF ENGINEERING
In
ELECTRONICS AND COMMUNICATION

Submitted by

AISHWARYA R.C.
AKHILA S RAO
AMITESH BHAGWAN

INTERNAL GUIDE
Mr. K.V MURALIDHAR
Assistant professor
Dept. of E & C
BIT, Bangalore

EXTERNAL GUIDE
Mr. ANANTH KRISHNA
Scientist/engineer SD
Environment test facility
ISAC, ISRO, Bangalore

HEAD OF THE DEPARTMENT:
Dr. C. S. SRIDHAR
Prof and Head, Dept. of E & C
BIT, Bangalore

DEPARTMENT OF ELECTRONICS AND COMMUNICATION
BANGALORE INTITUTE OF TECHNOLOGY
2008-2009

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
BANGALORE INTITUTE OF TECHNOLOGY
K. R. ROAD, V. V. PURAM, BANGALORE-560004

CERTIFICATE

It is certified that the project work entitled 'TEMPERATURE MEASUREMENT FOR SPACE APPLICATION USING CAPACITIVE SENSORS' is a bonafide work carried out by the following students

1. **AISHWARYA RAVICHANDER (1BI05EC002)**
2. **AKHILA S. RAO (1BI05EC005)**
3. **AMITESH BHAGWAN (1BI05EC008)**

in partial fulfillment of the requirement for the degree of Bachelor of Engineering in Electronics and Communication of the Visveswaraiah Technological University during the year 2008-2009. The project has been approved as it satisfies the academic requirement in respect to the project work prescribed for the Bachelor of Engineering degree.

Signature of the guide: Signature of HOD:

Mr. K.V. Muralidhar
Assistant professor,
Dept. of E & C,
BIT, Bangalore

Dr. C. S. Sridhar,
Professor and Head,
Dept. of E & C,
BIT, Bangalore

Name of the examiners:

1.

2.

Table of Contents

Abstract

In a diode due to a density gradient across the pn junction, holes will initially defuse to the right across the junction and electrons to the left. The positive holes which neutralized the acceptor ions near the junction in the p type silicon disappear as a result of combination with electrons. Similarly the neutralizing electrons in the n type silicon combine with holes that cross from the p junction, thus forming a region depleted of charges, the space charge region. Since this region is devoid of charges it acts as an insulator between two conducting slabs thus giving rise to junction capacitance.

The main power source for earth bound satellites is solar power which is received and converted to electrical energy by arrays of solar cells mounted on the surface of the satellites. Depending on their relative position to the sun these Low Earth Orbit satellites are typically exposed to temperatures ranging from +80 to -180 degree Celsius. Such drastic changes in temperature lead to dramatic changes in the current voltage characteristics of the solar cell. This occurs since the width of the space charge region varies inversely with the temperature of the solar cell causing the solar cell to have different resultant capacitance at each temperature.

This property of a solar cell capacitance to vary with temperature is made use of in this project to measure its temperature. This variation of capacitance is converted into the variation in frequency of a signal using a signal conditioning circuit. Finally a microcontroller is used to accept this signal as input, process it using various algorithms and compute the temperature value.

1 Introduction

1.1 About ISRO

Indian Space Research Organization was founded in 1972, when the government of India set up a space commission with a view to promote development and application of space technology and space science for the socio-economic benefit of the nation. The very foundation of the organization has to be attributed to the foresight and vision of Dr. Vikram. A. Sarabai who inspired Indian scientists to start developing satellites.

The prime objective of ISRO has been to develop space technology and its application to various national tasks. Since its inception, ISRO has established space systems like the INSAT for telecommunication, television broadcasting and meteorological services and the Indian remote sensing satellites (IRS) for resources monitoring and management. ISRO has also developed the satellite launch vehicles PSLV and GSLV to place these satellites in the required orbits.

ISRO is the prime agency charged with the responsibility laid down by the space commission and the development of space (DOS). It has made significant contributions in the areas of space, survey, engineering and technology. Main objective of the space program includes development of the satellites, launch vehicles, sounding rockets and associated ground systems.

ISRO consists of a number of centres geographically distributed all over the country. Each centre specializes in a specific area of space activity like some of the following.

- Vikram sarabhai space centre (VSSC)- ISRO's single largest facility, near Trivandrum providing the technology base for launcher and propulsion development

- Liquid propulsion system centre (LPSC)- Development branches in Bangalore and Trivandrum are supported by major test facilities at Mahendragiri for wide spectrum of liquid motors, from reaction control system thrusters to the 720kN Vikas and cryogenic engines
- ISRO Satellite Centre (ISAC)- ISRO's lead centre for design, fabrication and testing of science, technology and applications satellite
- ISRO inertial systems unit (IISU)- provides inertial systems and components for satellites and launches
- SATISH DAWAN SPACE CENTER- The ISRO's orbital launch site and largest solid motor production and taste facility
- ISRO telemetry, Tracking and command network (ITRAC) - Headquartered in Bangalore, ISTARC operates a network of ground station to provide TTC support for launcher and satellite operations.
- SPACE APPLICATION CENTRE (SAC)- Located at Ahmedabad, sac is ISRO's applications R & D center, including communications, remote sensing and geodesy
- Development and educational communications unit (DECU) at Ahmedabad

ISRO SATELLITE CENTRE (ISAC), Bangalore:

ISRO satellite centre (ISAC), Bangalore is one of the major research and development centre of Indian space research organization. It is the lead centre for satellite technology. The primary objective of ISAC is to develop and operationalize indigenous satellite and use space technology for the socio-economic development of the country. It was established in 1972 at Bangalore as ISSP (Indian Satellite Project) to build the nation's first satellite "ARAYABHATTA".

Over the years ISAC has planned and executed more than 35 satellite missions of ISRO. These missions represent a broad spectrum of satellite

technology. Having successfully developed and deployed state of art operationalized satellites for communications, meteorology and remote sensing, the centre is poised to a giant leap into the next century with advanced versions of satellite like microwave remote sensing satellites and direct broadcasting satellites. Projects are organized on an inter- group division basis with identified teams for the development

The activities at ISAC cover digital system, power system, communication and microwave system, spacecraft assembly integration and testing, structures, thermal spacecraft mechanism, control system, spacecraft mission planning and analysis, computers and information, system reliability and space physics. The facilities include fabrication and test facilities for satellite projects, laboratory for electro optic sensors (LEOS) works under the ISAC umbrella.

ISAC successfully designed and executed several satellite project of ISRO namely ARYABHATA, BHASKARA I, BHASKARA II, the three satellites of Rohini series, APPLE, SROSS- I, II AND CI and some of the latest are IRS-P4, TES, INSAT-3 series, KALPANA and GSAT-2.

1.2 Temperature measurement techniques currently in use

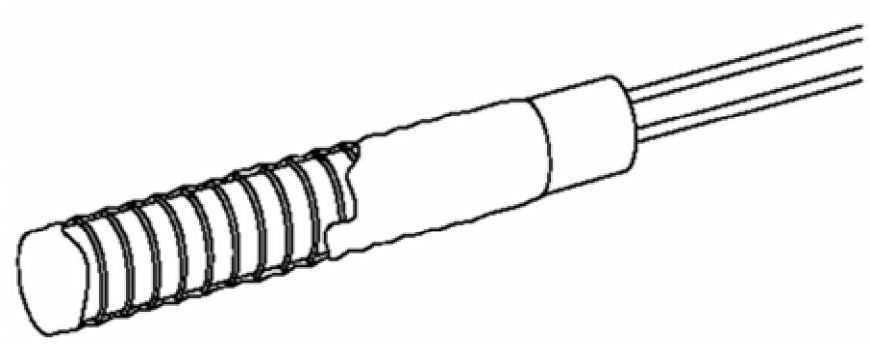
Currently, the technology used to measure the surface temperature of the solar panels is called **PLATINUM RESISTANCE THERMOMETRY.**

A platinum resistance thermometer (PRT) is a device which determines the temperature by measuring the electrical resistance of a piece of pure platinum wire.

The length and diameter of the platinum wire used in a thermometer are often chosen so that the resistance of the device at around 0 ºC is 100 ohms. Such a sensor is a called a PT100 sensor, and its resistance changes by approximately 0.4 ohms per degree Celsius. Using a typical 1 mA measuring current, at around 0 ºC a PT100 sensor would have a voltage drop of around 100 mV across its terminals and this would change by approximately 0.4 mV per degree Celsius, which thus makes sensitive thermometry available to anyone with a high resolution voltmeter or resistance bridge.

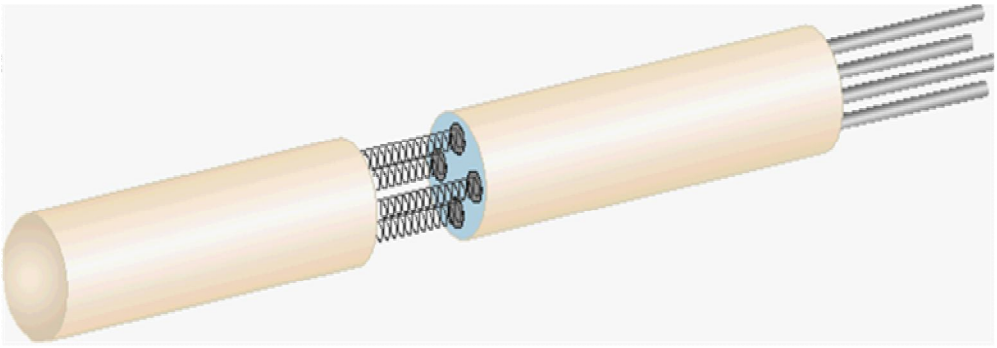
PRTs come in different form factors, two of which are implemented in satellites. These are:

(A) Wire-wound PRTs: These can have greater accuracy, especially for wide temperature ranges. The coil diameter provides a compromise between mechanical stability and allowing expansion of the wire to minimize strain and consequential drift.



(B) Coil Element PRTs: These are the more recent versions and have largely replaced wire wound elements in the industry. This design allows the wire coil to expand more freely over temperature while still provided the necessary

support for the coil. This design is similar to that of a SPRT, the primary standard which ITS-90 is based on, while still providing the durability necessary for an industrial process.



Working:

The electrical resistance of platinum increases linearly with absolute temperature.

The resistance of the platinum wire is measured by passing a current (AC or DC) through it and measuring the voltage with a suitable bridge or voltmeter, and the reading is converted to temperature using a calibration equation.

Drawbacks of this method:

- PRTs are less sensitive to small temperature changes
- They have a slower response time
- *Insulation Resistance:* Error caused by the inability to measure the actual resistance of element. Current leaks into or out of the circuit through the sheath, between the element leads, or the elements

2 Solar cells

2.1 The Basics

Solar cells, also called photovoltaic's (PV) by solar cell scientists, convert sunlight directly into electricity. Solar cells are often used to power calculators and watches. They are made of semiconducting materials similar to those used in computer chips. When sunlight is absorbed by these materials, the solar energy knocks electrons loose from their atoms, allowing the electrons to flow through the material to produce electricity. This process of converting light (photons) to electricity (voltage) is called the photovoltaic (PV) effect.

As we know from the properties of PN junction diodes when we place a layer of n-type silicon on a layer of p-type silicon, a barrier is created at the junction of the two materials due to recombination of holes and electrons leading to a region without any charge known as the space charge region. No charge carriers can cross this barrier if the diode is unbiased.

However when sunlight falls on the PN junction, one of these three things can happen depending on the band gap value of that particular cell:

The photon can pass straight through the silicon — this (generally) happens for lower energy photons,

The photon can reflect off the surface of the cell,

The photon can be absorbed by the silicon, if the photon energy is higher than the silicon band gap value. This generates an electron-hole pair and sometimes heat, depending on the band structure.

When a photon is absorbed, its energy is given to an electron in the crystal lattice. Usually this electron is in the valence band, and is tightly bound in covalent bonds between neighboring atoms, and hence unable to move far. The energy given to it by the photon "excites" it into the conduction band, where it is free to move around within the semiconductor. The covalent bond that the electron was previously a part of now has one fewer electron — this is known as a hole. The presence of a missing covalent bond allows the bonded electrons of neighboring atoms to move into the "hole," leaving another hole behind, and in this way a hole can move through the lattice. Thus, it can be said that photons absorbed in the semiconductor create mobile electron-hole pairs. This allows them to jump across the barrier to the n-type layer above and flow out around the circuit. Simply put the more light that shines, the more electrons jump up and the more current flows.

A photon need only have greater energy than that of the band gap in order to excite an electron from the valence band into the conduction band.

However, the solar frequency spectrum approximates a black body spectrum at ~6000 K, and as such, much of the solar radiation is composed of photons with energies greater than the band gap of silicon.

Solar cells are typically combined into modules that hold about 40 cells; about 10 of these modules are mounted in PV arrays that can measure up to several meters on a side. These flat-plate PV arrays can be mounted at a fixed angle facing south, or they can be mounted on a tracking device that follows the sun, allowing them to capture the most sunlight over the course of a day. About 10 to 20 PV arrays can provide enough power for a household; for large electric utility or industrial applications, hundreds of arrays can be interconnected to form a single, large PV system.

The performance of a solar cell is measured in terms of its efficiency at turning sunlight into electricity. Only sunlight of certain energies will work efficiently to create electricity, and much of it is reflected or absorbed by the materials that make up the cell. Because of this, a typical commercial solar cell has an efficiency of 15%—about one-sixth of the sunlight striking the cell generates electricity. Low efficiencies mean that larger arrays are needed, and that means higher cost. Improving solar cell efficiencies while holding down the cost per cell is an important goal of the PV industry, NREL researchers, and other U.S. Department of Energy (DOE) laboratories, and they have made significant progress. The first solar cells, built in the 1950s, had efficiencies of less than 4%.

The efficiency (n) of a solar cell is defined as the power P_{max} supplied by the cell at the maximum power point under standard test conditions, divided by the power of the radiation incident upon it.

2.2 More about the P-N junction

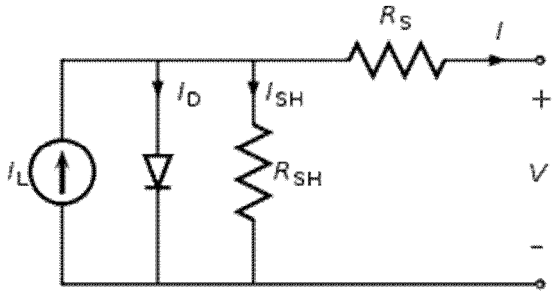
The most commonly known solar cell is configured as a large-area p-n junction made from silicon. As a simplification, one can imagine bringing a layer of n-type silicon into direct contact with a layer of p-type silicon. In practice, p-n junctions of silicon solar cells are not made in this way, but rather, by diffusing an n-type dopant into one side of a p-type wafer (or vice versa).

If a piece of p-type silicon is placed in intimate contact with a piece of n-type silicon, then a diffusion of electrons occurs from the region of high electron concentration (the n-type side of the junction) into the region of low electron concentration (p-type side of the junction). When the electrons diffuse across the p-n junction, they recombine with holes on the p-type side.

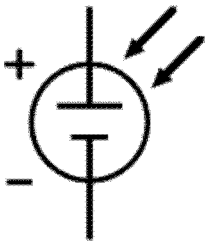
The diffusion of carriers does not happen indefinitely however, because of an electric field which is created by the imbalance of charge immediately on either side of the junction which this diffusion creates. The electric field established across the p-n junction creates a diode that promotes charge flow, known as drift current, that opposes and eventually balances out the diffusion of electron and holes. This region where electrons and holes have diffused across the junction is called the depletion region because it no longer contains any mobile charge carriers. It is also known as the "space charge region".

Ohmic metal-semiconductor contacts are made to both the n-type and p-type sides of the solar cell, and the electrodes connected to an external load. Electrons that are created on the n-type side, or have been "collected" by the junction and swept onto the n-type side, may travel through the wire, power the load, and continue through the wire until they reach the p-type semiconductor-metal contact. Here, they recombine with a hole that was either created as an electron-hole pair on the p-type side of the solar cell, or are swept across the junction from the n-type side after being created there.

The voltage measured is equal to the difference in the quasi Fermi levels of the minority carriers i.e. electrons in the p-type portion, and holes in the n-type portion.



The equivalent circuit of a solar cell



The schematic symbol of a solar cell

To understand the electronic behaviour of a solar cell, it is useful to create a model which is electrically equivalent, and is based on discrete electrical components whose behaviour is well known. An ideal solar cell may be

modeled by a current source in parallel with a diode; in practice no solar cell is ideal, so a shunt resistance and a series resistance component are added to the model. The resulting equivalent circuit of a solar cell is shown on the left. Also shown, on the right, is the schematic representation of a solar cell for use in circuit diagrams.

Characteristic equation

From the equivalent circuit it is evident that the current produced by the solar cell is equal to that produced by the current source, minus that which flows through the diode, minus that which flows through the shunt resistor.

$I = I_L - I_D - I_{SH}$

Where

I = output current (amperes)

I_L = photogenerated current (amperes)

I_D = diode current (amperes)

I_{SH} = shunt current (amperes)

The current flowing through these elements is governed by the voltage across them:

$V_j = V + IR_S$

Where

V_j = voltage across both diode and resistor R_{SH} (volts)

V = voltage across the output terminals (volts)

I = output current (amperes)

R_S = series resistance (Ω)

By the Shockley diode equation, the current diverted through the diode is:

$$I_D = I_0 \left\{ \exp\left[\frac{qV_j}{nkT}\right] - 1 \right\}$$

Where

I_0 = reverse saturation current (amperes)

n = diode ideality factor (1 for an ideal diode)

q = elementary charge

k = Boltzmann's constant

T = absolute temperature

For silicon at 25°C, $kT/q \approx 0.0259$ volts.

By Ohm's law, the current diverted through the shunt resistor is:

$$I_{SH} = \frac{V_j}{R_{SH}}$$

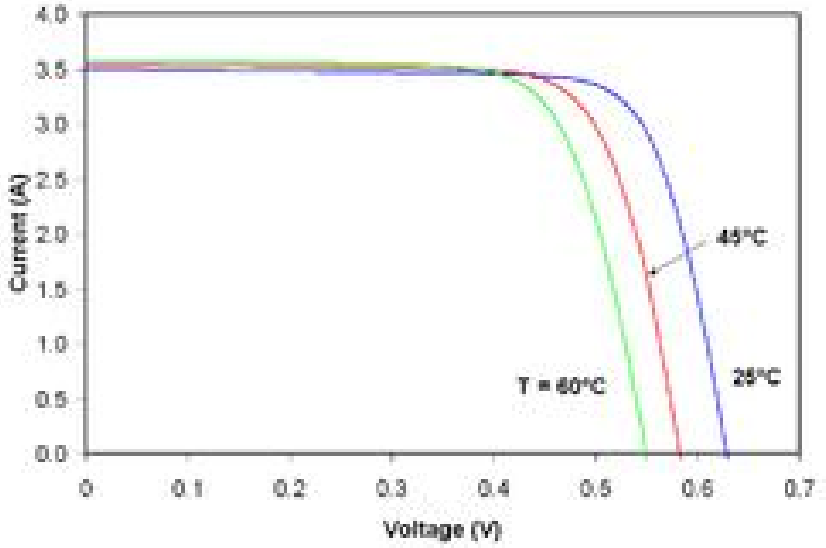
Where

R_{SH} = shunt resistance (Ω)

Substituting these into the first equation produces the characteristic equation of a solar cell, which relates solar cell parameters to the output current and voltage:

$$I = I_L - I_0 \left\{ \exp\left[\frac{q(V + IR_S)}{nkT} \right] - 1 \right\} - \frac{V + IR_S}{R_{SH}}$$

2.3 Response of solar cells to temperature variations



Effect of temperature on the current-voltage characteristics of a solar cell

Temperature affects the characteristic equation in two ways: directly, via T in the exponential term, and indirectly via its effect on I_0. (Strictly speaking, temperature affects all of the terms, but these two far more significantly than the others.) While increasing T reduces the magnitude of the exponent in the characteristic equation, the value of I_0 increases in proportion to exp$T$. The net effect is to reduce V_{OC} linearly with increasing temperature. The magnitude of this reduction is inversely proportional to V_{OC}; that is, cells with higher values of V_{OC} suffer smaller reductions in voltage with increasing temperature. For most crystalline silicon solar cells the reduction is about 0.50%/°C, though the rate for the highest-efficiency crystalline silicon cells is around 0.35%/°C. By way of comparison, the rate for amorphous silicon solar cells is 0.20-0.30%/°C, depending on how the cell is made.

The amount of photogenerated current I_L increases slightly with increasing temperature because of an increase in the number of thermally generated carriers in the cell. This effect is slight, however: about 0.065%/°C for crystalline silicon cells and 0.09% for amorphous silicon cells.

The overall effect of temperature on cell efficiency can be computed using these factors in combination with the characteristic equation. However, since the change in voltage is much stronger than the change in current, the overall effect on efficiency tends to be similar to that on voltage. Most crystalline silicon solar cells decline in efficiency by 0.50%/°C and most amorphous cells decline by 0.15-0.25%/°C. The figure to the right shows I-V curves that might typically be seen for a crystalline silicon solar cell at various temperatures.

3 The conditioning circuit

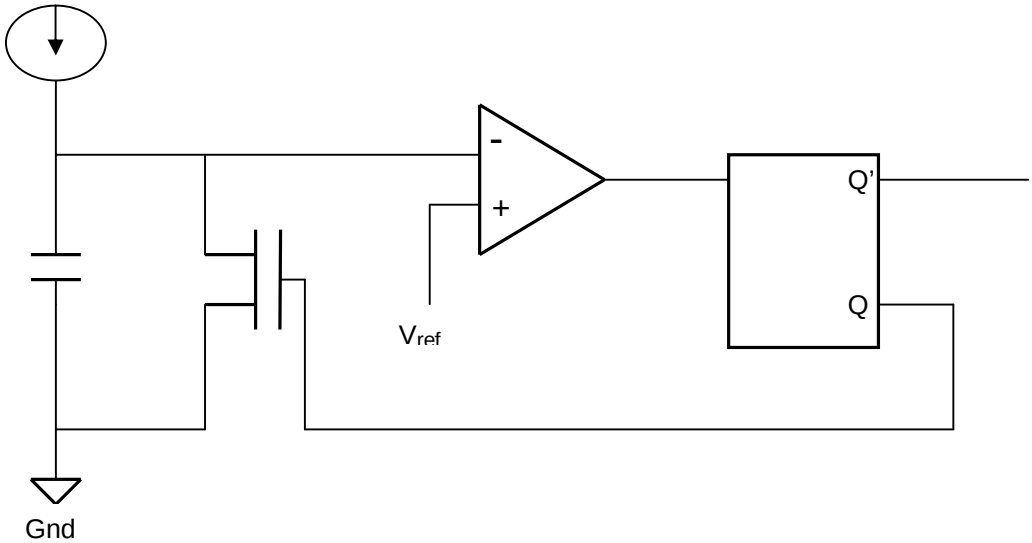
3.1 What it does

The essence of this project is the newly proposed method of measuring solar panel temperature, which is achieved by measuring the proportionally varying solar cell junction capacitance. This requires us to convert the variation in junction capacitance to a quantity that can be easily measured. This is exactly what the conditioning circuit does. It is called the conditioning circuit as it conditions a certain type of input into a required type of output.

What type of signal could a varying capacitance generate that would be a measure of its capacitance? A frequency modulated signal was our option. A frequency modulated signal is easy to measure hence a viable option. The solar cell junction capacitance can be made to charge and discharge to generate an output signal. The charging time of the capacitor depends on its capacitance hence an increase in capacitance (caused by an increase in temperature) would increase its charging time. Similarly a decrease in capacitance (caused by a decrease in temperature) would decrease its charging time. So here we can see how the capacitance of the solar cell and hence temperature are inversely proportional to the output signal frequency.

This charging and discharging of the capacitor is converted into a rectangular wave signal using a comparator. The rectangular wave can be directly fed into a microcontroller for frequency measurement.

We shall first attempt to understand the concept used in the conditioning circuit by studying a basic version that explains its functioning without considering design restraints.



Consider the basic circuit shown above and simultaneously observe the graph plots shown in the following section. The solar cell capacitance is charged by a constant current source. The comparator compares the capacitor voltage to a reference voltage $V_{ref.}$ When the voltage across the capacitor increases beyond V_{ref}, the comparator output goes high triggering the multivibrator which generates a high pulse on output pin Q and a low pulse on output pin Q'. The time duration of this pulse is preset by the multivibrator design. Q is connected to the gate of the FET and Q' is the frequency variant signal required.

Once the multivibrator is triggered the comparator output remains high for a very short duration of time, as the high pulse from the multivibrator turns ON the FET causing the capacitance to discharge through it. The FET is ON for the duration of the multivibrator pulse after which it turns OFF allowing the capacitor to charge again.

Let us now see how the output signal varies as the aforementioned activities happen. While the capacitor is charging, the comparator output is low and hence the output Q of the multivibrator is low and Q' high. Once the capacitor charges to V_{ref} the comparator output is high and hence multivibrator output Q' is a low pulse of fixed duration. Here we can see that Q' is high during charge time of the capacitor and depends on its capacitance. And during discharge time Q' generates a constant duration low pulse. Here it is to be noted that the circuit should be designed such that the discharge time is very small so it never exceeds the multivibrator pulse duration. This ensures that the output signal always has constant 'low time' duration and a variable 'high time' duration with the duration being proportional to the solar cell capacitance. Thus an increase in capacitance would mean longer 'high time' duration and hence a lower frequency signal. Similarly a decrease in capacitance would generate a higher frequency signal.

We know,

Temperature is directly proportional to cell capacitance.

Cell capacitance is inversely proportional to output signal frequency.

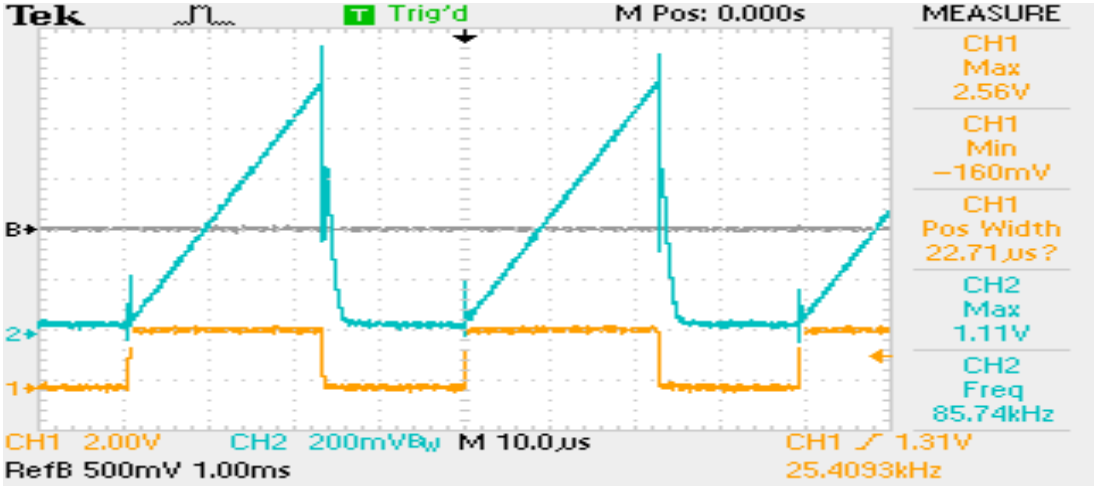
Hence,

Temperature is inversely proportional to output signal frequency.

3.2 Conditioning circuit response

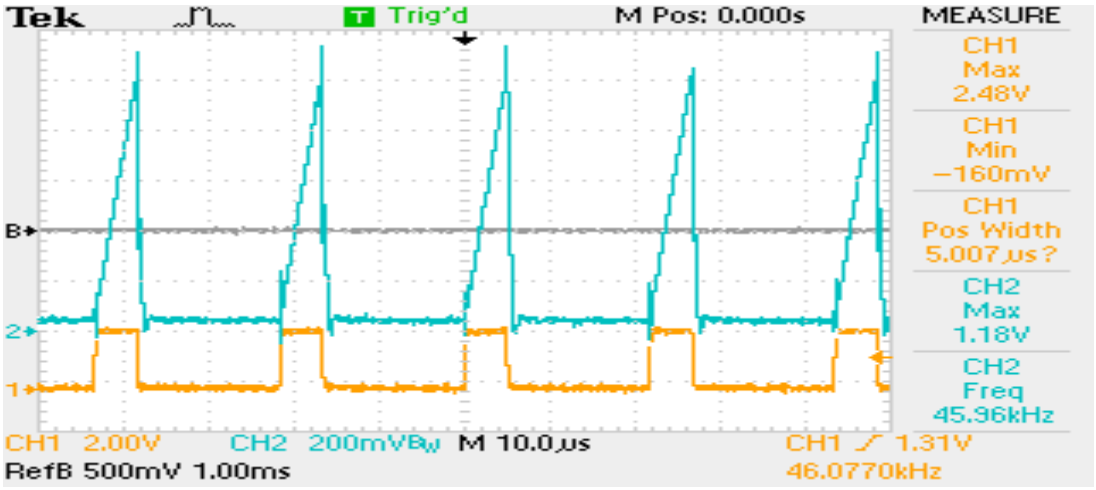
The voltage response at strategic points in the circuit has been shown here with corresponding CRO snapshots.

Each of the plots has been made in comparison with the conditioning signal final output (microcontroller input) for better perspective.



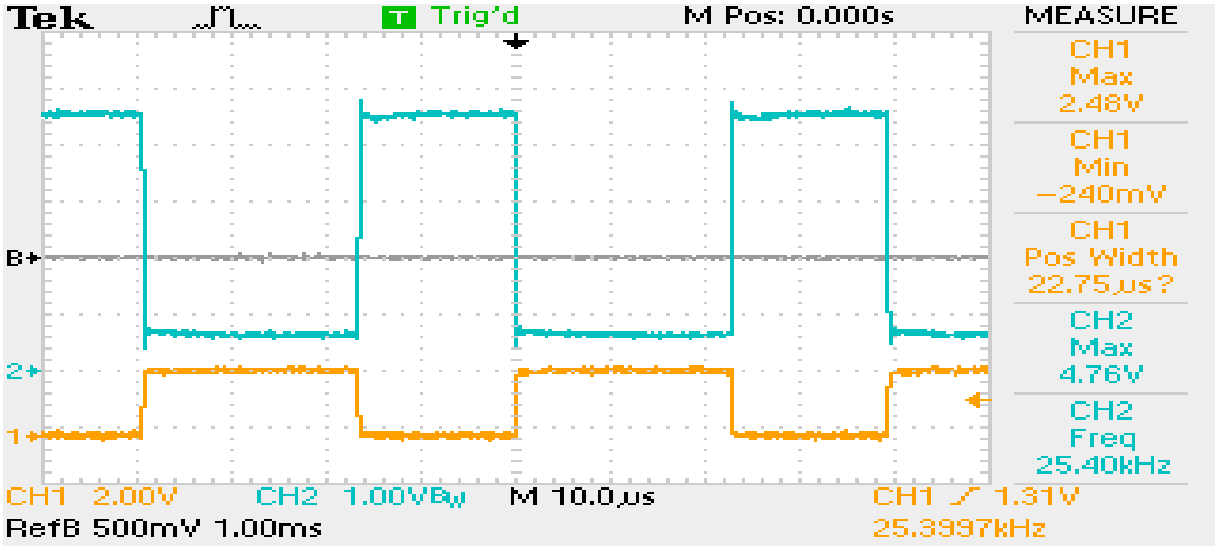
Channel 1 is at the input to microcontroller **Channel 2 is across the capacitor**

Plot at high cell capacitance corresponding to high temperature and low frequency

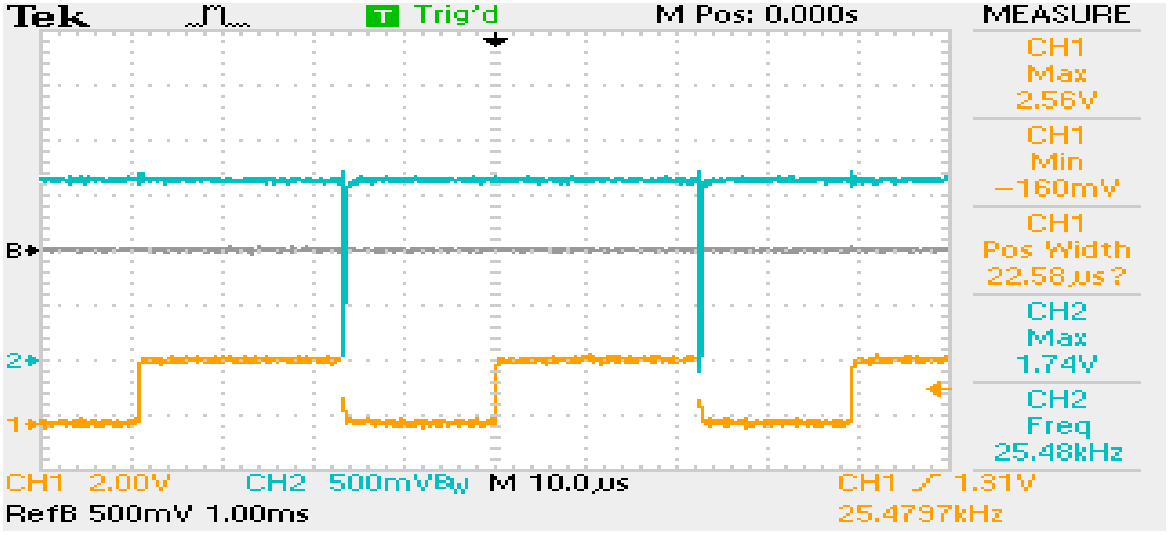


Channel 1 is at the input to microcontroller **Channel 2 is across the capacitor**

Plot at low cell capacitance corresponding to low temperature and high frequency



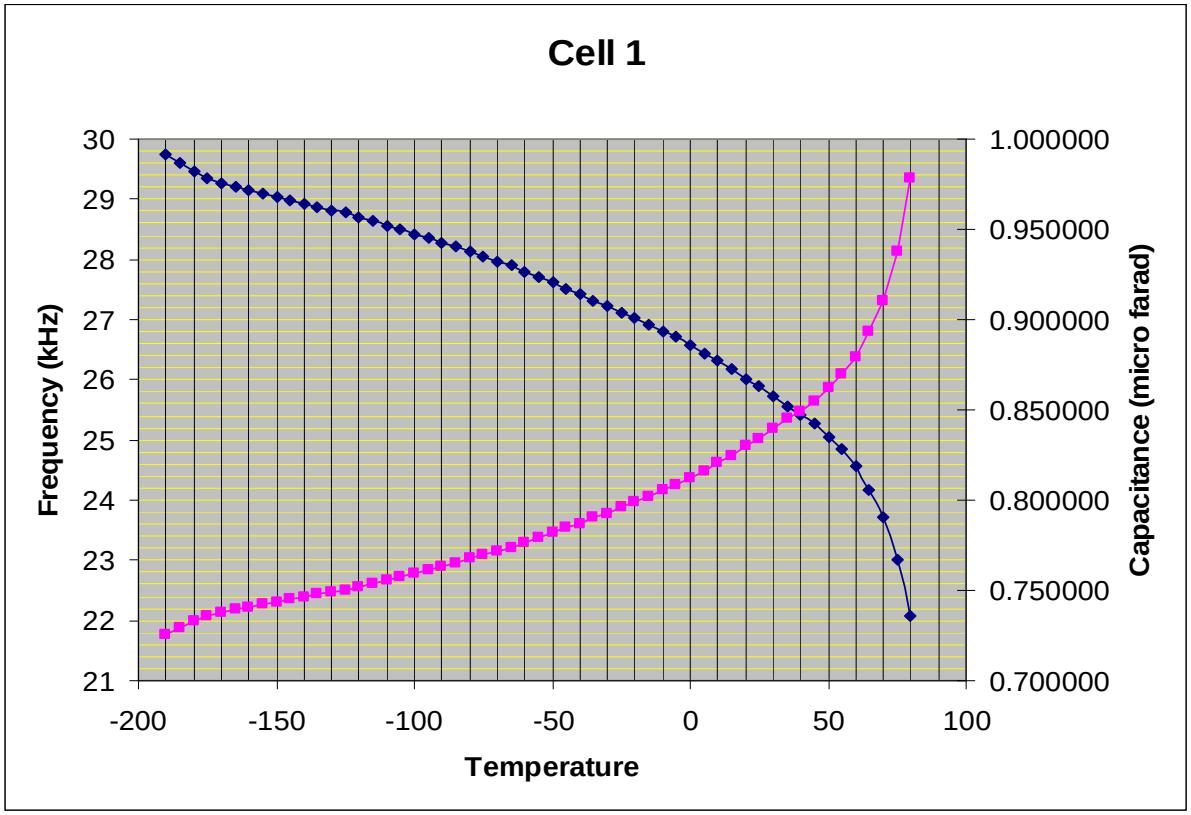
Channel 1 is at the input to microcontroller Channel 2 is at the input to the gate of FET



Channel 1 is at the input to microcontroller Channel 2 is at the output of comparator

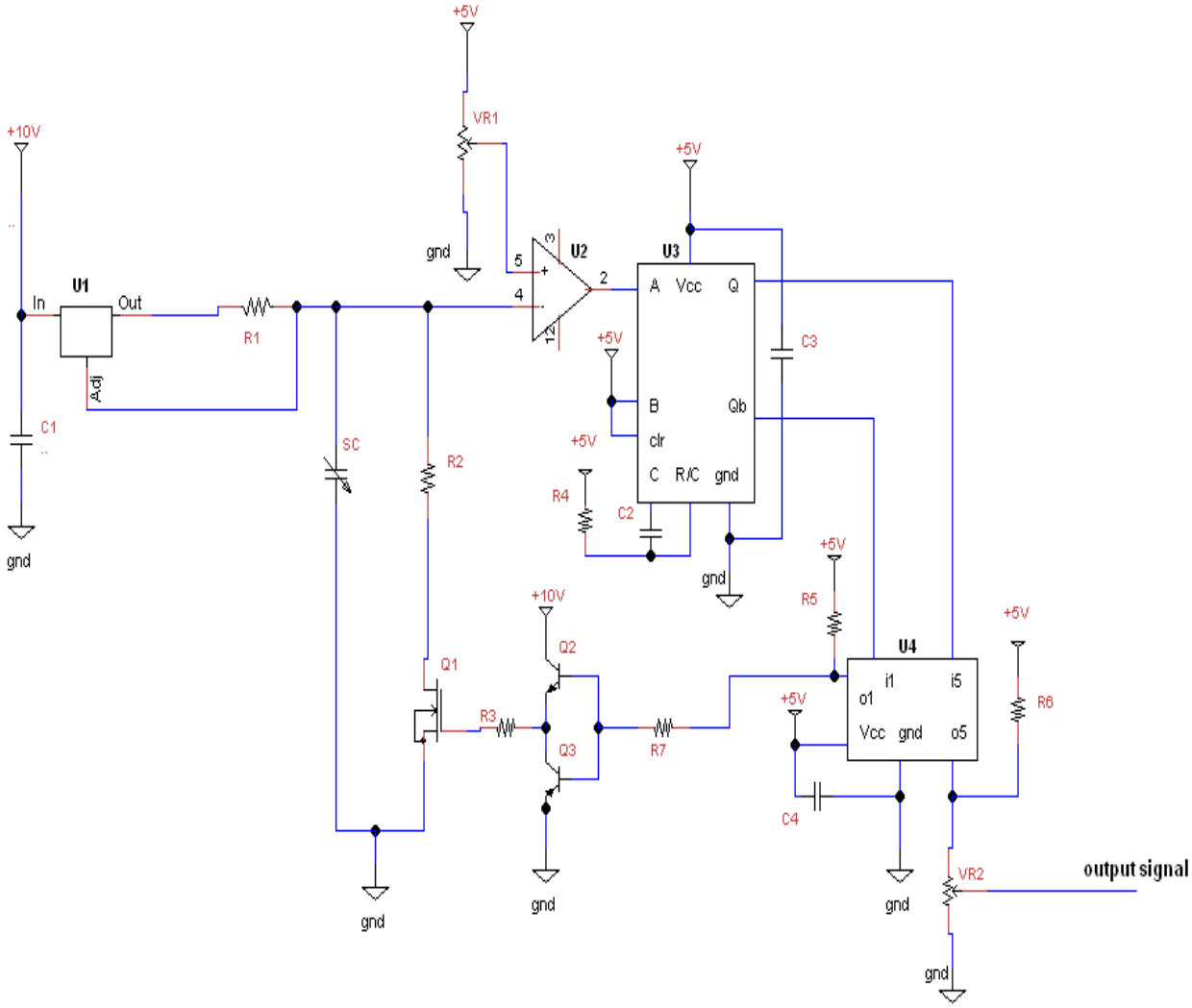
3.3 Test data of an actual cell

Solar cells are manufactured in several sizes and of different composition. Since each of these cells is constructed differently, they each have different responses to variations in temperature. The response to temperature variations of a particular test cell ‘Cell 1’ has been plotted below. The graph contains a plot of junction capacitance of solar cell versus temperature and a plot of frequency of conditioning circuit output signal (input to microcontroller) versus temperature. The proportionality relation between the three quantities is clear in the graphs.



3.4 Design

The actual implemented circuit based on the concept circuit in section 3.1 is given here. The pin diagrams and data sheets of the transistors and ICs used have been provided in the appendix. The implementation of the circuit in certain sections is consequential and not intentional.



4 Frequency measurement

4.1 The concept of measurement

Up till this section we talked about the hardware portion of the project. The hardware portion involved, first, describing the capacitance variation of solar cells as a measure for temperature variation. We discussed how a solar cell can be used in a circuit, as a capacitor, to produce a frequency modulated rectangular signal whose frequency is an indicator of temperature. What we need now is to measure the signal frequency to retrieve temperature information.

Frequency measurement is performed here using a microcontroller PIC16F877A. The choice of microcontroller and its features have been explained in the next section.

The microcontroller can use one of its timer/counters to measure frequency. A counter is activated, and the number of pulses received divided by the time period of measurement gives us the frequency of the signal.

4.2 Errors involved

It is very necessary to test and calculate the errors encountered in the system. An estimate of error margin encountered gives us an estimate of accuracy we can expect of our system.

Before the fine details of sampling time required and choice of counter can be made an error estimate has to be made, analyses and the option with least error has to be selected.

A estimate of maximum frequency that we will be measuring under the different situations of different cell characteristics or modifications in the conditioning circuit can be taken to be around 200kHz. Then the maximum time period for which a counter can count without overflow is given by,

If using an 8-bit counter: 256/200000 = 1. 28ms

If using a 16-bit counter: 65535/200000 = 327. 675ms

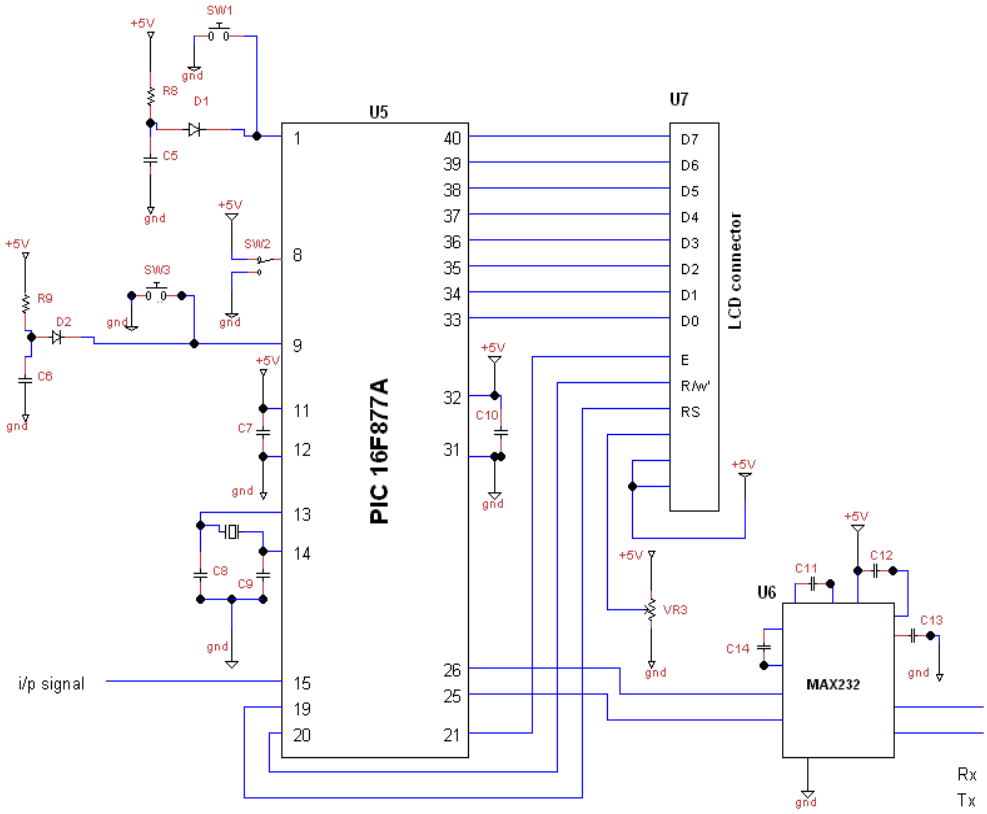
Using an 8-bit counter for a time period of 1ms gives us a huge error margin of 1000Hz since the measured count will have to be multiplied by a factor of 1000 to get the actual frequency. This seems unacceptable for the level of accuracy required in space applications.

Using a 16-bit counter for a time period of 250ms gives us an error margin of

4Hz as the multiplication factor here is only 4, which is very small and acceptable. Therefore for our application we use a measuring time of 250ms so that frequencies up to 262.14 kHz can be measured with an error of at most 4Hz.

4.3 The microcontroller set-up

The setup of the microcontroller along with necessary connections like reset switch, biasing components is as shown below:



5 The Microcontroller – PIC16F877A

5.1 Features

The PIC16F877A is a member of the PIC16F87X family. We have chosen this microcontroller over its peers due to the obvious advantages it possesses, specifically for our purpose.

The PIC16F877A comes in a 40 pin DIP package.

One main reason for its popularity is that it is very cheap. Apart from that it is also very easy to assemble. Additional components needed to make this IC work are just a 5V power supply adapter, a 20MHz crystal oscillator and 2 units of 22pF capacitors.

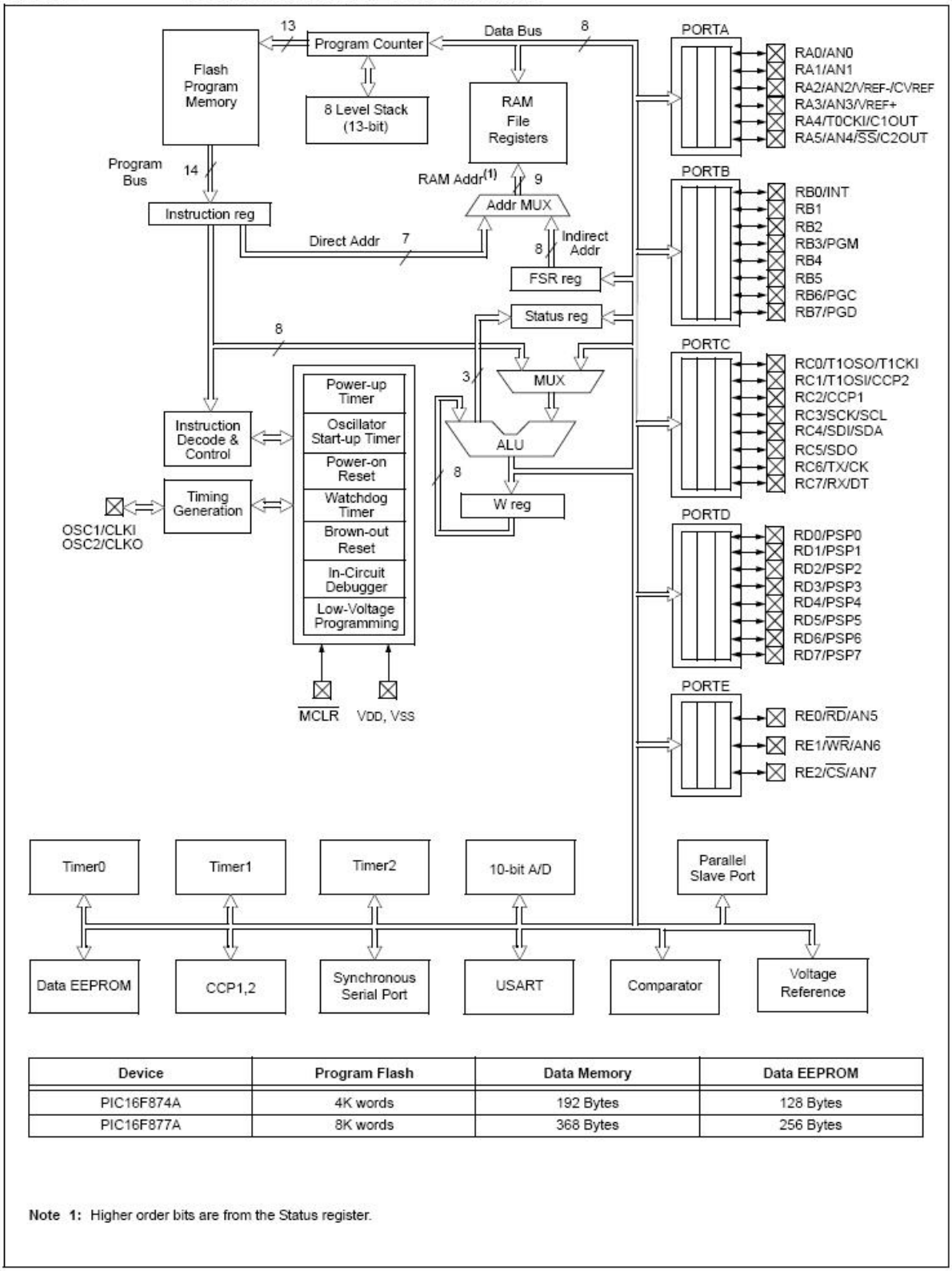
Features of PIC16F877A:

- High-performance RISC CPU
  - Lead-free:RoHS-compliant
  - Operating speed:20Mhz, 200ns instruction cycle
  - Operating voltage:4.0-5.5volts
  - Industrial temperature range(-40 to +85 degrees)
  - 15 Interrupt sources
  - 35 single word instructions
  - All single-cycle instructions except for program branches
- Special Microcontroller Features
  - Flash memory:14.3KB(8192 words)
  - Data SRAM:368 bytes
  - Data EEPROM:256 bytes
  - Self-reprogrammable under software control
  - In-circuit serial programming via two pins
  - Watchdog timer with on-chip RC oscillator
  - Programmable code protection
  - Power-saving code protection
  - Selectable oscillator options
  - In-circuit debug via two pins

- Peripheral Features
    - 33 I/O pins : 5 I/O ports
    - Timer0:8-bit timer/counter with 8-bit prescaler
    - Timer1:16-bit timer/counter with prescaler
    - Timer2:8-bit timer/counter with 8-bit period register, prescaler and postscaler
    - Two capture, compare, PWM modules
    - Synchronous serial port with two modes
    - USART/SCI with 9-bit address detection
    - Parrallel slave port
    - Brown-out detection circuitry for brown-out reset
- Analog Features
    - 10-bit, 8-channel A/D converter
    - Brown-out reset
    - Analog comparable module

The detailed internal block diagram of the 16F877A is given in the next page.

FIGURE PIC16F874A/877A BLOCK DIAGRAM



Device	Program Flash	Data Memory	Data EEPROM
PIC16F874A	4K words	192 Bytes	128 Bytes
PIC16F877A	8K words	368 Bytes	256 Bytes

Note 1: Higher order bits are from the Status register.

Another important point worth mentioning here is the large FLASH EEPROM size. The 16F877A comes with an 8k word FLASH EEPROM. This was much needed for our project, to store the frequency-temperature translation table, which was the heart of our program.

5.2 The programming tools

These programming tools are the special applications used during the course of development of this project.

5.2.1 PIC C editor and compiler

The program code was written using this editor and also compiled using the same.

The PIC-C compiler supports coding in embedded C as well as in ASM.

The advantage of using embedded C lies in the power of the several built-in functions. The advantage of Assembly language programming on the other hand is in the complete control it offers to the user.

To harness the advantages of both, we have written certain parts in ASM and others in Embedded C.

FEATURES:

The CCS C Compiler features provide ample functions for the development needs of this project, including: standard C pre-processor directives, operators & statements, built-in libraries supporting all chips, MPLAB® IDE integration, source code drivers, automatic linking for multiple code pages and built in IDE & simulation environment.

More importantly, there is also Access to hardware features from C, which includes:

- 1, 8, 16 and 32 bit integer types and 32 bit floating point.
- Standard one bit type (Short Int) permits the compiler to generate very efficient Bit oriented code.
- #BIT and #BYTE will allow C variables to be placed at absolute addresses to map registers to C variables.
- Bit Arrays
- Fixed Point Decimal
- Constants (including strings and arrays) are saved in program memory.
- Flexible Handling Of Constant Data
- Variable Length Constant Strings
- Address mod Capability To Create User Defined Address Spaces In Memory Device

ASM was used wherever it proved to be the best option and Embedded C was used in places where it simplified the job.

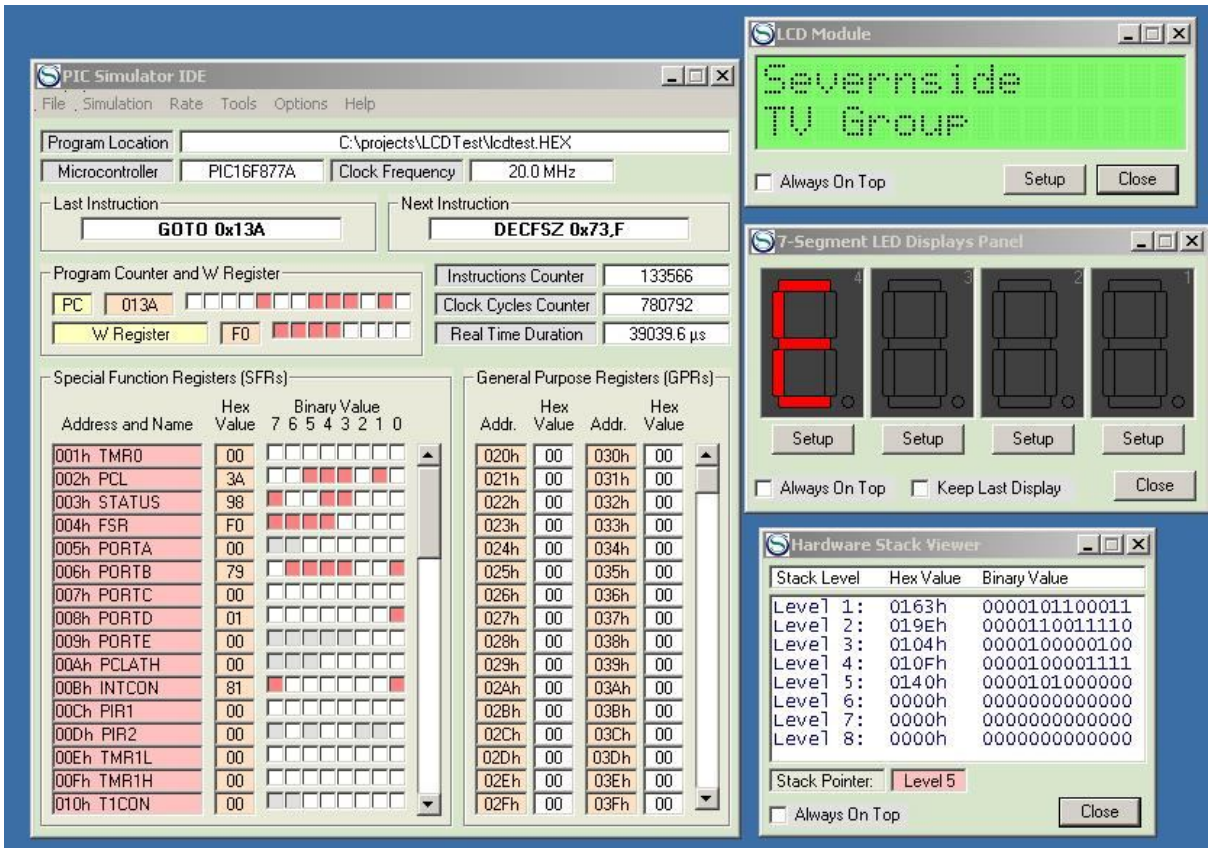
5.2.2 Oshon Simulator (Simulation tool - Version 6.61)

This was invaluable in the initial stages, where learning to program the PIC MC and studying its registers in detail was the main objective.

The peripheral function and timer/counter view settings were the most used tools. Other features include

- Main simulation interface showing internal microcontroller architecture,
- FLASH program memory editor, EEPROM data memory editor, hardware stack viewer,
- Microcontroller pinout interface for simulation of digital I/O and analog inputs,
- Variable simulation rate, simulation statistics,
- Breakpoints manager for code debugging with breakpoints support,

Below is a snapshot of the **oshonsoft simulation tool**

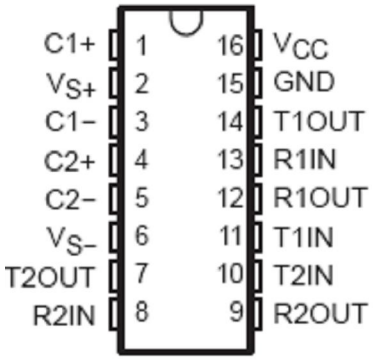


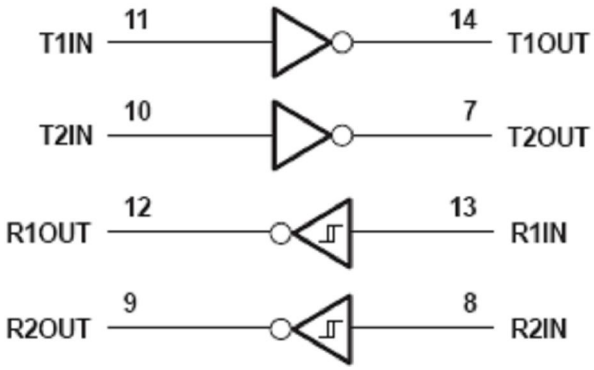
5.2.3 RS232

RS232 is an asynchronous serial communications protocol, widely used on computers. Asynchronous means it doesn't have any separate synchronising clock signal, so it has to synchronise itself to the incoming data - it does this by the use of START' and 'STOP' pulses. The signal itself is slightly unusual for computers, as rather than the normal 0V to 5V range, it uses +12V to -12V - this is done to improve reliability, and greatly increases the available range it can work over - it isn't necessary to provide this exact voltage swing, and you can actually use the PIC's 0V to 5V voltage swing with a couple of resistors to make a simple RS232 interface which will usually work well, but isn't guaranteed to work with all serial ports. For this reason we have used the MAX232 chip, this is a chip specially designed to interface between 5V

Logic levels and the +12V/-12V of RS232 - it generates the +12V/-12V internally using capacitor charge pumps, and includes four converters, two transmit and two receive.

MAX 232 IC



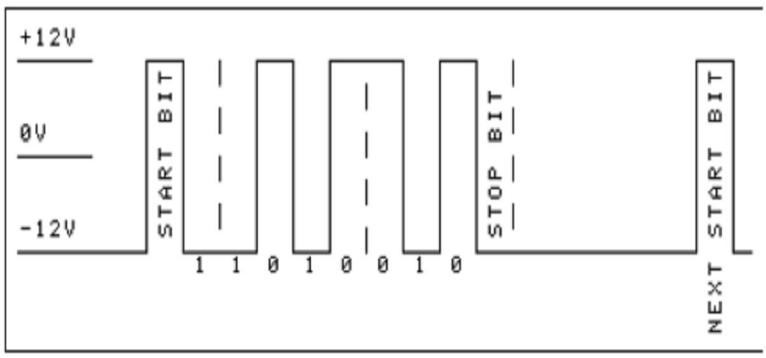


The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply TIA/EIA-232-F voltage levels from a single 5-V supply. Each receiver converts TIA/EIA-232-F inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V, a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into TIA/EIA-232-F levels.

The MAX232 converts the 0-5V signal pulses to their equivalent +/-12V RS232 pulses and vice-versa. If you put 5V on the T1IN pin, you will see 12V on the T1OUT pin. This is how you pass data out to the computer. If you press a key in hyperterminal, a signal is sent down the line to the R1IN pin where the 12V signal coming

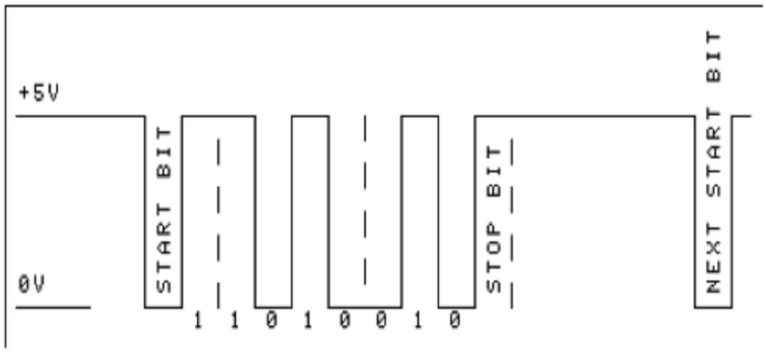
from the computer is converted to a 0/5V signal coming out of R1OUT - a signal that the PIC understands.

The real magic is how an IC powered with 5V can generate +/- 12V signals. This is done with the external capacitors known as charge pumps.



This is an example of a signal on an RS232 line, initially it sits at -12V, known as the 'STOP CONDITION', this condition lasts until a signal is sent. To send a signal we first need to let the receiving device know we are starting to send data, to do this we set the line to +12V, this is called the 'START BIT' - the receiving device is waiting for this to happen, once it does it then gets ready to read the next 9 bits of data (eight data bits and one stop bit).

This is the identical signal as it leaves (or enters) the PIC pin, as the MAX232 inverts the signal this looks to be inverted, but is actually the correct way up - RS232 logic levels are inverted compared to normal levels.



Code

The simple C program that is used on the receiving (computer) end is as follows:

It mostly uses built C functions that querry the serial port and display the data received.

```
#include <bios.h>
#include <conio.h>
#define COM1       0
#define DATA_READY 0x100
#define SETTINGS ( 0x80 | 0x02 | 0x00 | 0x00)
int main(void)
{
  int in, out, status;
```

```
    bioscom(0, SETTINGS, COM1); /*initialize the port*/
    cprintf("Data sent to you:  ");
    while (1)
    {
      status = bioscom(3, 0, COM1); /*wait until get a data*/
      if (status & DATA_READY)
          if ((out = bioscom(2, 0, COM1) & 0x7F) != 0)  /*input a data*/
            putch(out);
          if (kbhit())
          {
            if ((in = getch()) == 27)   /* ASCII of Esc*/
              break;
            bioscom(1, in, COM1);   /*output a data*/
          }
    }
    return 0;
}
```

6 Programming the Microcontroller

The microcontroller used and its desired features have been discussed in the previous section. With that information divulged, we can proceed to discus how the microcontroller is programmed to achieve the desire output. The input to the microcontroller is the frequency modulated signal from the conditioning circuit and the output should be a display of corresponding temperature. To achieve this we must compute the frequency of the input signal calculate its corresponding temperature and display it. These are the functions that are to be performed by the microcontroller.

Once the features and architecture of a microcontroller have been studied it becomes easy to convert a conceptual algorithm into an executable program code.

This section discusses in detail the algorithms employed and the final executable program code written to achieve the desired output.

A more detailed look into the input, the desired output and the PIC16F877A features gives us indications of how to go about achieving the output.

The input is a frequency modulated rectangular wave signal. Its frequency can be found using a counter to count the number of received pulses in a known duration of time.

Once frequency of the signal is known we need to compare it with our existing look-up table consisting of frequencies against corresponding temperature to find a best fit frequency and correspondingly the temperature. This data is then to be displayed on a display device.

With this very basic algorithm in mind we proceed to construct a more detailed algorithm in the coming sections. The first section; temperature measurement algorithm is a method in which the look-up table of frequency versus temperature is input into the microcontroller memory along with the program code. The second method; calibration algorithm is designed to externally take in the values to be entered into the look-up table before being able measure temperature. The relevance of each algorithm is discussed in their respective sections.

6.1 Temperature measurement algorithm

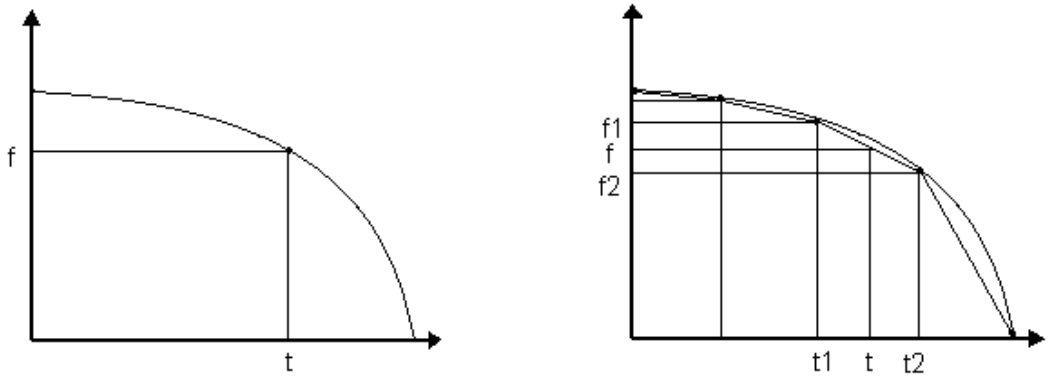
The temperature measurement algorithm accepts the frequency modulated input signal, measures its frequency, uses a look-up table to find the best fit temperature and displays the temperature.

Details of how the microcontroller is used to do the above and why the method we used was chosen is described here

- Input signal frequency measurement can be done using one of the 3 timers PIC16F877A processes. Of the 3 timers Timer 0 and Timer 2 are 8 bit timers, Timer1 is a 16 bit timer/counter and hence was selected to accommodate a higher frequency without the complication of handling count overflow. Timer 1 is thus configured as a 16-bit counter that counts external pulses.
- The time period of sampling of the input signal is fixed at 250ms. With a sampling period of 250ms ($1/4^{th}$ s) the maximum measurable input signal frequency is around 262 kHz (FFh * 4). This value of maximum measurable frequency gives the program adequate flexibility to accommodate the use of different solar cells with different junction capacitance variation responses.
- The look-up table used in the program consists of a frequency array and a temperature array of equal sizes with each frequency element corresponding to the input frequency received from the conditioning circuit for the value of the corresponding temperature element at the same position. The frequency array elements range from ~~~~~~~~Hz to ~~~~~Hz and the corresponding temperature values range from +80 to -180 degree centigrade. The temperature array consists of 53 elements with a temperature spacing of 5 degrees between consecutive elements.
- Using only the look-up table a temperature accuracy of 5 degrees can be achieved. To improve accuracy the method of interpolation is used between two array elements.
- The microcontroller is then programmed to convert the temperature data into its ASCII equivalent and display on an LCD.
- The program performs measurement every 1s and updates the display.

6.1.1 Interpolation

The most important concept of the temperature measurement algorithm is the concept of interpolation. Since the difference between consecutive temperature values in the look-up table are sufficiently small, the curve joining these two points on a frequency versus temperature graph can be assumed to be a straight line. Once this assumption is made he 4^{th} degree equation curve relating frequency and temperature can be reduced to a series of short straight lines. When the input signal frequency is between two consecutive look-up table frequency elements, the two look-up table points are assumed to be connected by a straight line and the temperature value is found by using the input signal frequency in the straight line equation defined by the end points.



Y1 and Y2 are values from the frequency look-up table between which 'F' exists. X1 and X2 are the corresponding values from the temperature look-up table. 'F' is the measured frequency of input signal. And 'T' is its corresponding temperature.

Assuming (X1, Y1) and (X2, Y2) are the two points joined by a straight line, an equation defining the straight line is given by

(y - Y1) / (x - X1) = (Y1 -Y2) / (X1 - X2)

Or

y = [(Y1 -Y2) / (X1 - X2)] * (x - X1) + Y1

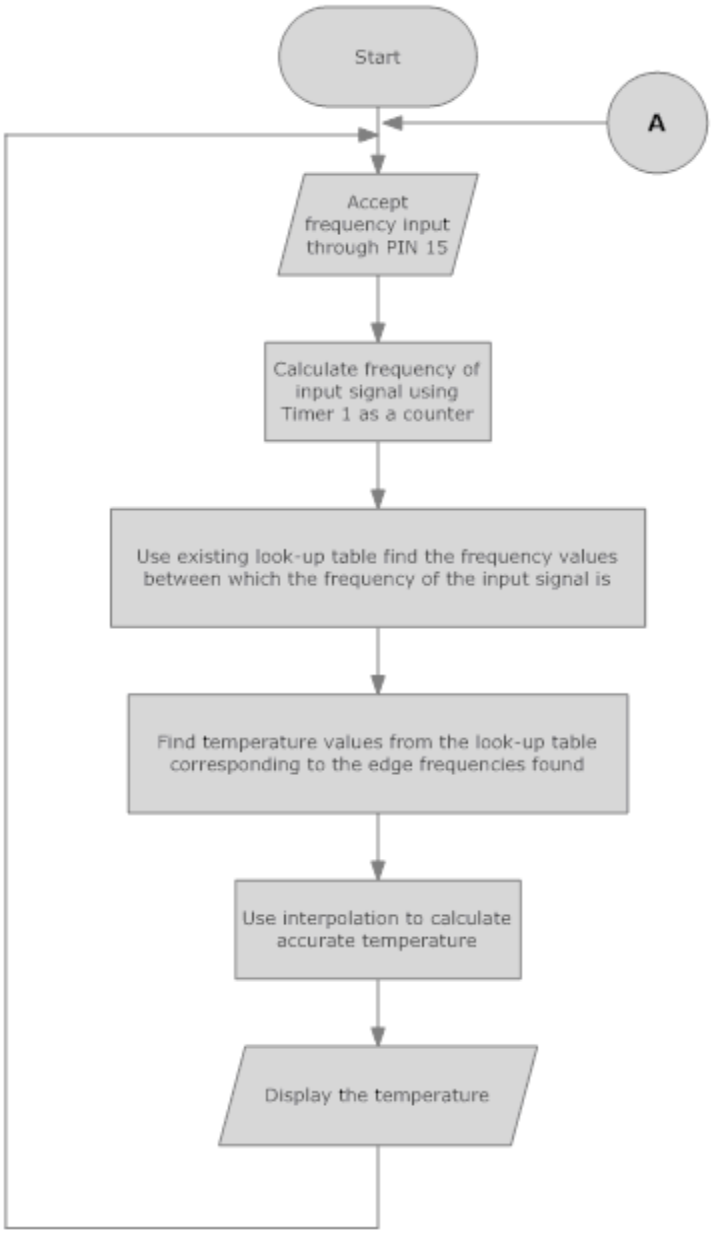
Using this equation to find temperature (T) for an input frequency (F)

T = [(X1 - X2) / (Y1 - Y2)] * (F - Y1) + X1

This method of calculating temperature gives better accuracy than using the look-up table alone.

6.1.2 Basic flowchart

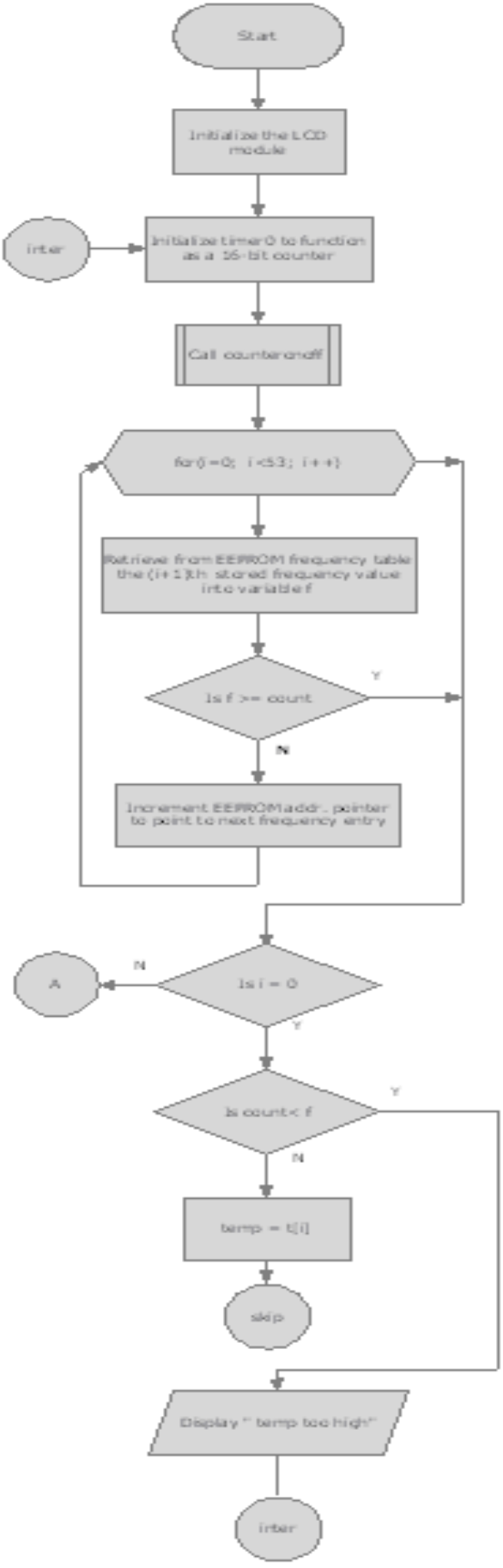
Here is a basic flow diagram of the temperature measurement algorithm.

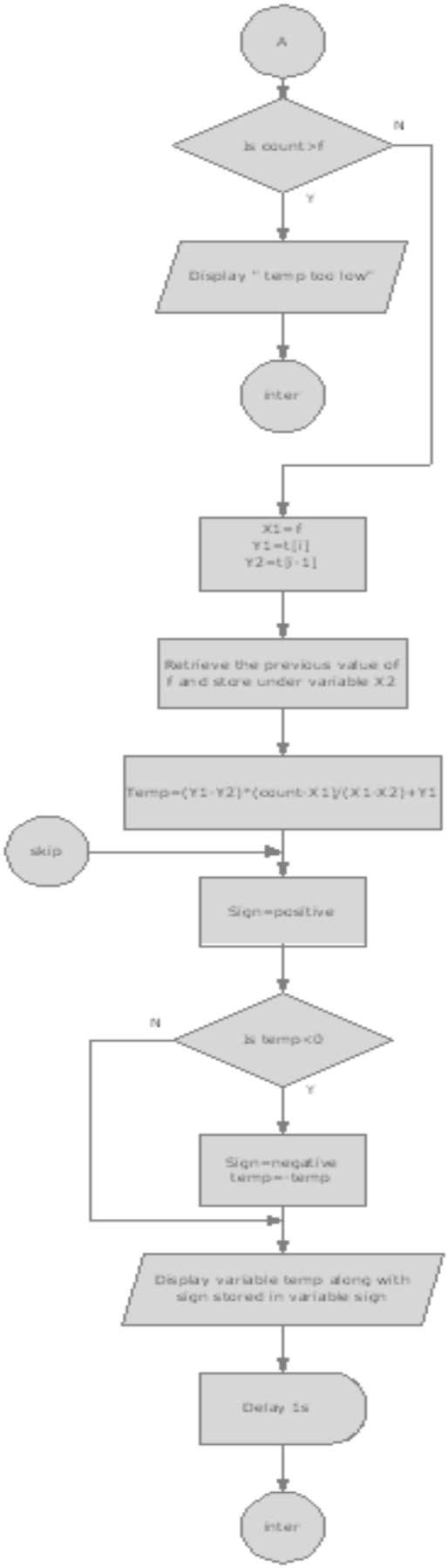


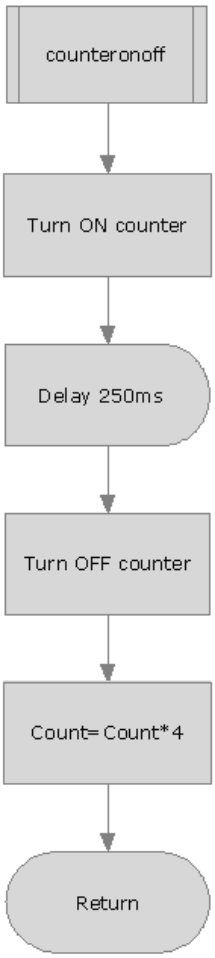
Connector ‘A’ connects to this flowchart from another flowchart in the coming section.

6.1.3 Advanced flowchart

A more detailed advanced flowchart of the basic flow chart is given here. This flowchart is in complete synchrony with the flow of the program code.




A
N
Y
Display " temp too low"
inter
X1=f
Y1=t[i]
Y2=t[i-1]
Retrieve the previous value of f and store under variable X2
Temp=(Y1-Y2)*(count-X1)/(X1-X2)+Y1
skip
Sign=positive
N
Y
Sign=negative
temp=-temp
Display variable temp along with sign stored in variable sign
Delay 1s
inter



6.1.4 Code

```
#include "D:\bitiiiiiiii\testprogs_2603\back\interpolation23.h"
        #use delay(6144000)


int8 du,x,td,dt,dh,b,i,j,sign,y,addr=0;

const signed int16 t[53]={80,75,70,65,60,55,50,45,40,35,30,25,20,15,10,5,0,-
5,-10,-15,-20,-25,-30,-35,-40,-45,-50,-55,-60,-65,-70,-75,-80,-85,-90,-95,-100,
-105,-110,-115,-120,-125,-130,-135,-140,-145,-150,-155,-160,-165,-170,-175,-
180};

signed int16 x1,x2;
int32 count,y1,y2;
```

```
const int32 f[53]=
{195288,194994,194599,194221,193900,193552,193253,192889,
192541,192262,191967,191645,191250,190893,190508,190129,189768,189
436,189037,188663,188216,187783,187366,186898,186369,185787,185227,
184643,183970,183338,182666,181955,181217,180488,179772,178800,177
894,177018,176109,175217,174116,173117,172400,171348,170363,169267,
167735,166235,164532,161913,158707,153853, 147164};
int8 tempint,tempdec;
float temp;

void main()
{
delay_ms(15);
#asm
call lcdinit

sense:
call sensedisp
#endasm
 delay_ms(2000);
 #asm
        movlw 0x01
        call command
        movlw 0x80
        call command
        #endasm

inter:
for(i=0;i<53;i++)
{
 y1=f[i];
y2=f[i-1];
x1=t[i];
x2=t[i-1];
```

```
 if(f[i]>=count)
        break;
}



if(i==0)
{
 if(count>=f)
  {
   temp=t[i];
   goto skip;
  }
  #asm
  call high
#endasm
goto inter;
}

if (count>f)
 {
 #asm
 call low
#endasm
goto inter;
}

temp=(((float)x1-(float)x2)/((float)y1-(float)y2))*((float)count-
(float)y1)+(float)x1;
skip:        sign=0;
     if(temp<0)
     {
     sign=1;
     temp=-temp;
     }
```

```
#asm
movlw 0x80//1stline ctrl=80 to 8f; 2nd line=c0to7f//
call command
movlw 'T'
call disp1
movlw 'E' // data to disp//
call disp1
movlw 'M' // data to disp//
call disp1
movlw 'P' // data to disp//
call disp1
movlw '=' // data to disp//
call disp1
movlw ' '
#endasm
if(sign==0)
     {
      #asm
      movlw '+'
      call disp1
      #endasm
     }
else
     {
      #asm
      movlw '-'
      call disp1
      #endasm
     }
tempint=temp;
tempdec=(temp-(float)tempint)*100;
x=tempint;
#asm
call hba3
```

```
call disp3
movlw '.'
call disp1
#endasm
x=tempdec;
#asm
call hba2
call disp2
#endasm
delay_ms(1000);
goto inter;

 lcdinit:
       movlw 0x00
        movwf 0x86  //tris_b=output//
        movlw 0x01
        movwf 0x87  //tris_c=output(except c.0,which is signal ip)//
        movlw 0x00
        movwf 0x88 //tris_d =output//
        movlw 0x38 //func set(8 bit data,2 line disp,5X8dots)//
        movwf 0x06 //out to port b//
        bcf 0x08,0 // portd,bit0=RS pin//
        bcf 0x08,1 //portd,bit1=r/w pin//
        bsf 0x08,2 //portd,bit2=enable pin//
        nop
        nop
        nop
        nop
        nop
        bcf 0x08,2
#endasm
delay_ms(5) ; // 5 ms//
#asm
movlw 0x38  //func set(8 bit data,2 line disp,5X8dots)//
```

```
movwf 0x06 //out to port b//
        bcf 0x08,0 // portd,bit1=RS pin//
        bcf 0x08,1 //portd,bit2=r/w pin//
        bsf 0x08,2 //portd,bit3=enable pin//
        nop
        nop
        nop
        nop
        nop
        bcf 0x08,2
#endasm
delay_us(100) ; // delay//
#asm
movlw 0x38  //func set(8 bit data,2 line disp,5X8dots)//
call command
movlw 0x0c   //disp ON,cursor disp,no cursor blink//
call command
movlw 0x01   //clr disp//
call command
movlw 0x06   //entry mode-set=incrs the pointr right continuously//
call command
movlw 0x80//1stline ctrl=80 to 8f; 2nd line=c0to7f//
call command
return


command:

        call ready
        movwf 0x06 //out to port b//
        bcf 0x08,0 // portd,bit1=RS pin//
        bcf 0x08,1 //portd,bit2=r/w pin//
        bsf 0x08,2 //portd,bit3=enable pin//
        nop
```

```
        nop
        nop
        nop
        nop
        bcf 0x08,2
        return

disp1: call ready
        bsf 0x08,0//rs//
        bcf 0x08,1//rw//
        bsf 0x08,2//enable//
        nop
        nop
        nop
        nop
        nop
        movwf 0x06
        nop
        bcf 0x08,2
        return

ready:
      bsf 0x86,7 // make pin 7 port b input//
      bcf 0x08,0 //rs//
      bsf 0x08,1 // r/w //
back2:  bcf 0x08,2 // enable//
      nop
      nop
      nop
      nop
      nop
      bsf 0x08,2
      btfsc 0x06,7 // busy pin//
      goto back2
```

```
        bcf 0x86,7
        return



countersetting://counter setting//
                movlw 0x00
                movwf 0x0e//set tmr1l 00//
                movwf 0x0f//tmr1h 00//
                bsf 0x10,1//t1con tmr1cs make it counter//
                bcf 0x10,2//t1con t1sync make it sync//
                bcf 0x10,3//t1con oscillator shut down//
                bcf 0x10,4
                bcf 0x10,5
                bsf 0x87,0//tris_c=for setting pinC,0 as input//
                return

hba2:
        #endasm
        du=0;
        dt=0;

        du=x%10;
        du=du+48;
        dt=x/10;
        dt=dt+48;
        #asm
        return

disp2:
     call ready
                bsf 0x08,0//rs//
                bcf 0x08,1//rw//
                bsf 0x08,2//enable//
```

```
                nop
                nop
                nop
                nop
         #endasm
        output_b(dt);
#asm
        nop
        nop
        nop
        nop
         bcf 0x08,2
                call ready
                bsf 0x08,0//rs//
                bcf 0x08,1//rw//
                bsf 0x08,2//enable//
                nop
                nop
                nop
                nop
         #endasm
        output_b(du);
          #asm
        nop
        nop
        nop
        nop
         bcf 0x08,2
           return


hba3: #endasm
        du=0;
        dt=0;
```

```
        dh=0;
        du=x%10;
        du=du+48;
        x=x/10;
        dt=x%10;
        dt=dt+48;
        dh=x/10;
        dh=dh+48;
        #asm
        return

disp3:
         call ready
             bsf 0x08,0//rs//
             bcf 0x08,1//rw//
             bsf 0x08,2//enable//
             nop
             nop
             nop
             nop
     #endasm
    output_b(dh);
        #asm
    nop
    nop
    nop
    nop
     bcf 0x08,2
        call ready
             bsf 0x08,0//rs//
             bcf 0x08,1//rw//
             bsf 0x08,2//enable//
             nop
             nop
```

```
            nop
            nop
        #endasm
        output_b(dt);
            #asm
        nop
        nop
        nop
        nop
         bcf 0x08,2
            call ready
                bsf 0x08,0//rs//
                bcf 0x08,1//rw//
                bsf 0x08,2//enable//
                nop
                nop
                nop
                nop
        #endasm
        output_b(du);
            #asm
        nop
        nop
        nop
        nop
         bcf 0x08,2
            return

sensedisp:

          movlw 0x80//1stline ctrl=80 to 8f; 2nd line=c0to7f//
          call command
          movlw 'S' // data to disp//
          call disp1
```

```
        movlw 'E' // data to disp//
        call disp1
        movlw 'N' // data to disp//
        call disp1
        movlw 'S' // data to disp//
        call disp1
        movlw 'E' // data to disp//
        call disp1
        movlw ' ' // data to disp//
        call disp1
        movlw 'M'// data to disp//
        call disp1
        movlw 'O' // data to disp//
        call disp1
        movlw 'D'// data to disp//
        call disp1
        movlw 'E' // data to disp//
        call disp1
         return


countonoff:

            movlw 0x00
            movwf 0x0e//set tmr1l 00//
            movwf 0x0f//tmr1h 00//
            bsf 0x10,0//t1con= turn on ctr1//
            #endasm
            delay_ms(249);
            delay_us(999);
            #asm
  bcf 0x10,0//t1con= trn off ctr1//
 #endasm
            count=get_timer1();
```

```
        count*=4;
        #asm
        return

high:   movlw 0x01//clr//
        call command
        movlw 0x80//1stline ctrl=80 to 8f; 2nd line=c0to7f//
        call command
        movlw 'T' // data to disp//
        call disp1
        movlw 'E' // data to disp//
        call disp1
        movlw 'M' // data to disp//
        call disp1
        movlw 'P' // data to disp//
        call disp1
        movlw ' ' // data to disp//
        call disp1
        movlw 'T' // data to disp//
        call disp1
        movlw 'O'// data to disp//
        call disp1
        movlw 'O' // data to disp//
        call disp1
        movlw ' '// data to disp//
        call disp1
        movlw 'H' // data to disp//
        call disp1
        movlw 'I'//1stline ctrl=80 to 8f; 2nd line=c0to7f//
        call disp1
        movlw 'G'//1stline ctrl=80 to 8f; 2nd line=c0to7f//
        call disp1
        movlw 'H'//1stline ctrl=80 to 8f; 2nd line=c0to7f//
        call disp1
```

```
        #endasm
        delay_ms(4000);
        #asm
        return

low:    movlw 0x01//clr//
        call command
        movlw 0x80//1stline ctrl=80 to 8f; 2nd line=c0to7f//
        call command
        movlw 'T' // data to disp//
        call disp1
        movlw 'E' // data to disp//
        call disp1
        movlw 'M' // data to disp//
        call disp1
        movlw 'P' // data to disp//
        call disp1
        movlw ' ' // data to disp//
        call disp1
        movlw 'T' // data to disp//
        call disp1
        movlw 'O'// data to disp//
        call disp1
        movlw 'O' // data to disp//
        call disp1
        movlw ' '// data to disp//
        call disp1
        movlw 'L' // data to disp//
        call disp1
        movlw 'O'//1stline ctrl=80 to 8f; 2nd line=c0to7f//
        call disp1
        movlw 'W'//1stline ctrl=80 to 8f; 2nd line=c0to7f//
        call disp1
        #endasm
```

```
        delay_ms(4000);
        #asm
        return

         #endasm
}
```

6.2 Calibration algorithm

The calibration algorithm is an extended version of the temperature measurement algorithm. Calibration algorithm has an extra portion in the program that allows the program to externally accept the frequency look-up table array by physically varying the temperature of the solar cell sensor and entering into the look-up table the frequency measured for each 5 degree variation in temperature.

The setup required for calibration process consists of the general temperature measurement setup along with a thermocouple thermometer placed near the solar cell sensor and a mechanism to expose the sensors to the required temperature range.

The detailed procedure of calibration can be explained in the following steps

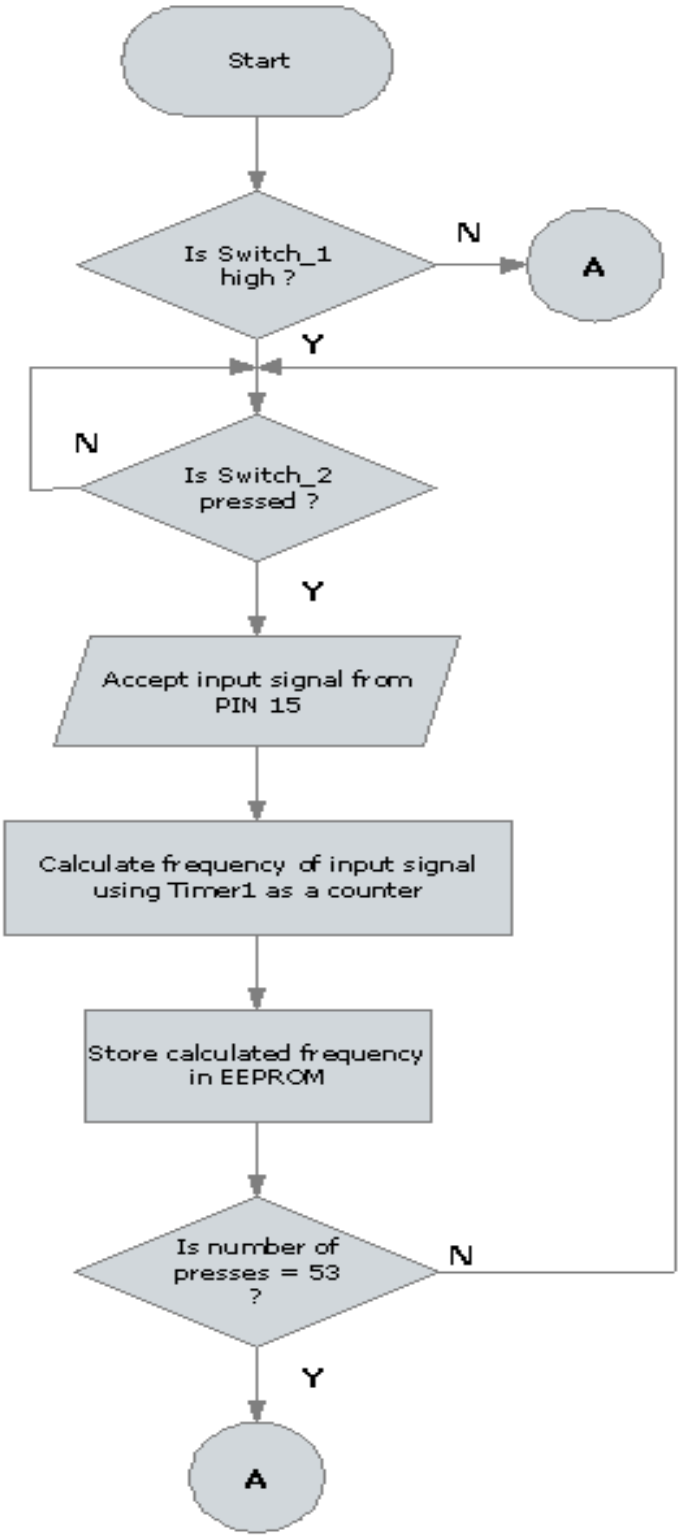
- A switch cwitch_1 is used to function in either calibration mode or simple temperature sensing mode.

- A switch switch_2 is used to indicate to the microcontroller to measure frequency of input signal at required temperature and store in appropriate location in the look-up table.
- When the microcontroller is reset it checks switch_1 to see which mode to perform. If calibration mode is selected, then it displays to the user the temperature at which switch_2 is to be pressed. The temperature of the sensors must then be brought to the mentioned temperature manually, checked with the thermocouple display by the user and switch_2 pressed at the mentioned temperature. This value of measured frequency gets stored in the look-up table against the temperature it was measured at. This is done for as many entries as the look-up table requires by slowly varying the temperature of the sensors.
- Once the entire look-up table is filled by the microcontroller it automatically enters the temperature sensing or measurement mode and performs what the previous algorithm performed.

A measurement system needs to be calibrated only once. The reason to include the calibration option in our program is because all solar cells are not manufactured identically. To eliminate error due to slight variation in manufacturing of these solar cells, calibration is performed on ground for the particular batch of solar cells being used as sensors for temperature. This ensures reliability.

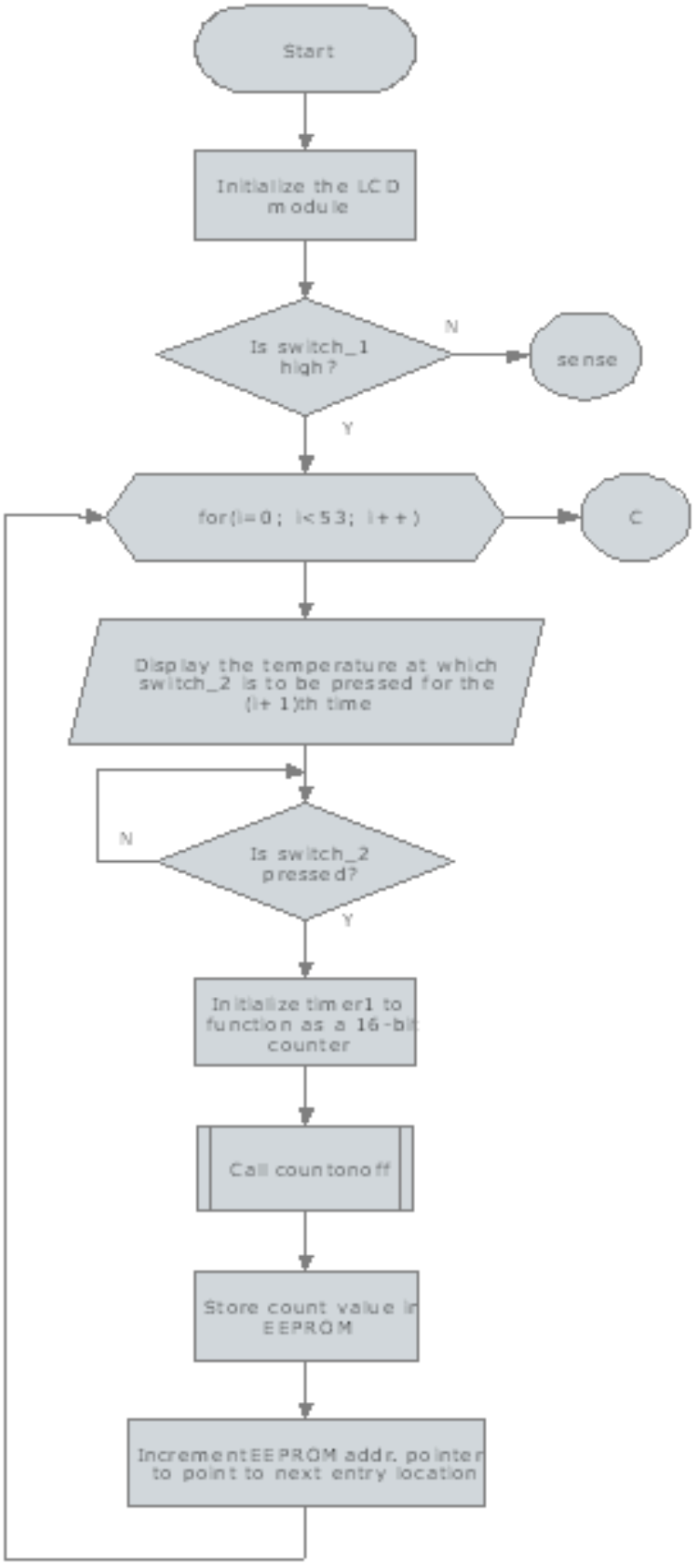
6.2.1 Basic flowchart

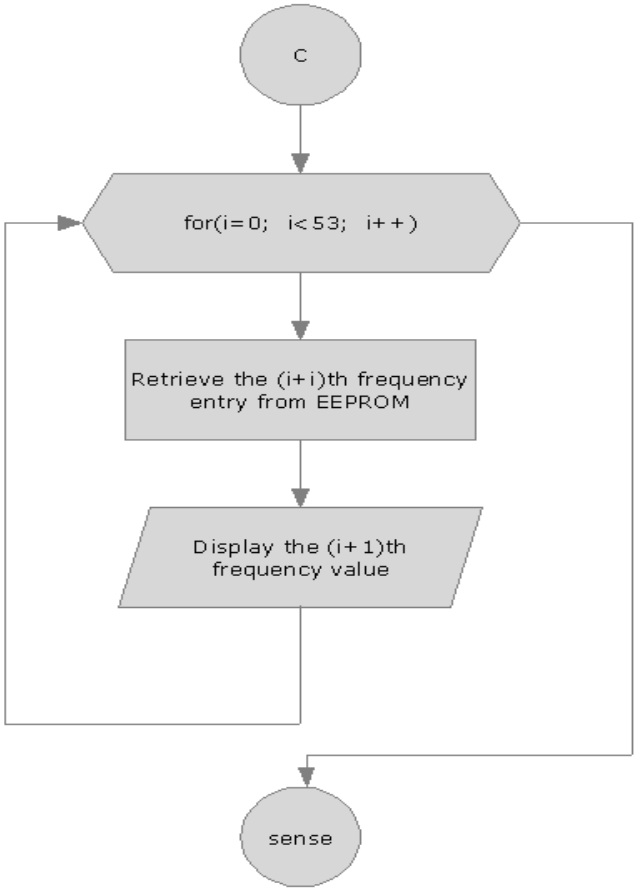
Here is a basic flow diagram of the calibration algorithm.



6.2.2 Advanced flowchart

A more detailed advanced flowchart of the basic flow chart is given here. This flowchart is in complete synchrony with the flow of the program code.





6.2.3 Code

```
#include "D:\bitiiiiiiii\testprogs_2603\back\newcalibapr29.h"
 #use delay(11059200)


int8 du,x,td,dt,dh,b,i,j,sign,y,addr=0;
const signed int16 t[53]={80,75,70,65,60,55,50,45,40,35,30,25,20,15,10,5,0,-
5,-10,-15,-20,-25,-30,-35,-40,-45,-50,-55,-60,-65,-70,-75,-80,-85,-90,-95,-100,
-105,-110,-115,-120,-125,-130,-135,-140,-145,-150,-155,-160,-165,-170,-175,-
180};
int32 a;
signed int16 x1,x2;
int32 count,y1,y2;
int32 f;
int8 tempint,tempdec;
float temp;

void main()
{
```

```
delay_ms(15);
#asm
call lcdinit
//polling switches//
movlw 0x03  // set e0 and e1 as input//
movwf 0x89 //tris e//
btfss 0x09,0 //MODE SELECTION SW1//
goto sense

//calibration//
call calidisp
#endasm
delay_ms(2000);
#asm
movlw 0x01
call command
movlw 0x80 // data to disp//
call command
movlw 'P' // data to disp//
call disp1
movlw 'R' // data to disp//
call disp1
movlw 'E' // data to disp//
call disp1
movlw 'S' // data to disp//
call disp1
movlw 'S' // data to disp//
call disp1
movlw ' ' // data to disp//
call disp1
movlw 'a' // data to disp//
call disp1
movlw 't' // data to disp//
call disp1
movlw ' ' // data to disp//
call disp1
movlw 'T' // data to disp//
call disp1
movlw '=' // data to disp//
call disp1
```

```
#endasm
for(i=0;i<53;i++)
{
#asm
movlw 0x8b// initializing ptr//
call command
#endasm
if (t[i]<0)
{
#asm
movlw '-'
call disp1
#endasm
td=-t[i];
}
else
{
#asm
movlw '+'
call disp1
#endasm
td=t[i];
}
x=td;
#asm
call hba3
call disp3
notpressed:  btfss 0x09,1            // switch 2 //
                goto notpressed
          #endasm
          delay_ms(1000);// switch debounce //
          #asm
          call countersetting
          call countonoff
          #endasm
         y=count%100;
           write_eeprom(addr,y);
           addr=addr+1;
           count=count/100;
```

```
            y=count%100;
              write_eeprom(addr,y);
              addr=addr+1;

              count=count/100;

              y=count%100;
              write_eeprom(addr,y);
              addr=addr+1;
              count=count/100;

              y=count%100;
              write_eeprom(addr,y);
              addr=addr+1;
}
// nowto display all the stored nos//
addr=3;
            for(i=0;i<53;i++)
            {
            #asm
            movlw 0x01   //clr disp//
            call command
            movlw 0x80   //clr disp//
            call command
            #endasm
            delay_ms(1000);
           for(j=0;j<4;j++)
            {
            x=read_eeprom(addr);
            #asm
            call hba2
            call disp2
            #endasm
            addr-=1;
            }
            addr=addr+8;
            delay_ms(2000);
}
#asm
sense:
```

```
call sensedisp
#endasm
 delay_ms(2000);
 #asm
        movlw 0x01
        call command
        movlw 0x80
        call command
        #endasm

inter:
        addr=3;
        #asm
        call countersetting
        call countonoff
        #endasm
        for(i=0;i<53;i++)
        {
        a=read_eeprom(addr);
        addr=addr-1;
        a=a*100;
        b=read_eeprom(addr);
        addr=addr-1;
        a=a+b;
        a=a*100;
        b=read_eeprom(addr);
        addr=addr-1;
        a=a+b;
        a=a*100;
        b=read_eeprom(addr);
        a=a+b;
        f=a;
 if(f>=count)
        break;
addr+=7;
        }

if(i==0)
{
 if(count>=f)
```

```
            {
             temp=t[i];
             goto skip;
            }
            #asm
            call high
           #endasm
           goto inter;
           }

           if (count>f)
            {
            #asm
            call low
           #endasm
           goto inter;
           }
            y1=f;
            x1=t[i];
            x2=t[i-1];
            addr=addr-1;
            a=read_eeprom(addr);
           addr=addr-1;
           a=a*100;
           b=read_eeprom(addr);
           addr=addr-1;
           a=a+b;
           a=a*100;
           b=read_eeprom(addr);
           addr=addr-1;
           a=a+b;
           a=a*100;
           b=read_eeprom(addr);
           a=a+b;
           y2=a;
           temp=(((float)x1-(float)x2)/((float)y1-(float)y2))*((float)count-
(float)y1)+(float)x1;
skip:      sign=0;
           if(temp<0)
           {
           sign=1;
```

```
    temp=-temp;
    }
#asm
movlw 0x80//1stline ctrl=80 to 8f; 2nd line=c0to7f//
call command
movlw 'T'
call disp1
movlw 'E' // data to disp//
call disp1
movlw 'M' // data to disp//
call disp1
movlw 'P' // data to disp//
call disp1
movlw '=' // data to disp//
call disp1
movlw ' '
#endasm
if(sign==0)
    {
     #asm
     movlw '+'
     call disp1
     #endasm
    }
else
    {
     #asm
     movlw '-'
     call disp1
     #endasm
    }
tempint=temp;
tempdec=(temp-(float)tempint)*100;
x=tempint;
#asm
call hba3
call disp3
movlw '.'
call disp1
#endasm
```

```
x=tempdec;
#asm
call hba2
call disp2
#endasm
delay_ms(1000);
#asm
goto inter



lcdinit:
        movlw 0x00
         movwf 0x86  //tris_b=output//
         movlw 0x01
         movwf 0x87  //tris_c=output(except c.0,which is signal ip)//
         movlw 0x00
         movwf 0x88 //tris_d =output//
         movlw 0x38 //func set(8 bit data,2 line disp,5X8dots)//
         movwf 0x06 //out to port b//
         bcf 0x08,0 // portd,bit0=RS pin//
         bcf 0x08,1 //portd,bit1=r/w pin//
         bsf 0x08,2 //portd,bit2=enable pin//
         nop
         nop
         nop
         nop
         nop
         bcf 0x08,2
#endasm
delay_ms(5) ; // 5 ms//
#asm
movlw 0x38  //func set(8 bit data,2 line disp,5X8dots)//
movwf 0x06 //out to port b//
        bcf 0x08,0 // portd,bit1=RS pin//
        bcf 0x08,1 //portd,bit2=r/w pin//
        bsf 0x08,2 //portd,bit3=enable pin//
        nop
        nop
        nop
```

```
        nop
        nop
        bcf 0x08,2
#endasm
delay_us(100) ; // delay//
#asm
movlw 0x38  //func set(8 bit data,2 line disp,5X8dots)//
call command
movlw 0x0c   //disp ON,cursor disp,no cursor blink//
call command
movlw 0x01   //clr disp//
call command
movlw 0x06   //entry mode-set=incrs the pointr right continuously//
call command
movlw 0x80//1stline ctrl=80 to 8f; 2nd line=c0to7f//
call command
return


command:

        call ready
        movwf 0x06 //out to port b//
        bcf 0x08,0 // portd,bit1=RS pin//
        bcf 0x08,1 //portd,bit2=r/w pin//
        bsf 0x08,2 //portd,bit3=enable pin//
        nop
        nop
        nop
        nop
        nop
        bcf 0x08,2
        return
disp1: call ready
         bsf 0x08,0//rs//
         bcf 0x08,1//rw//
         bsf 0x08,2//enable//
         nop
         nop
         nop
```

```
        nop
        nop
        movwf 0x06
        nop
        bcf 0x08,2
        return



ready:
    bsf 0x86,7 // make pin 7 port b input//
    bcf 0x08,0 //rs//
    bsf 0x08,1 // r/w //
back2:  bcf 0x08,2 // enable//
    nop
    nop
    nop
    nop
    nop
    bsf 0x08,2
    btfsc 0x06,7 // busy pin//
    goto back2
    bcf 0x86,7
    return

calidisp:

        movlw 0x80//1stline ctrl=80 to 8f; 2nd line=c0to7f//
        call command
        movlw 'C' // data to disp//
        call disp1
        movlw 'A' // data to disp//
        call disp1
        movlw 'L' // data to disp//
        call disp1
        movlw 'I' // data to disp//
        call disp1
        movlw 'B' // data to disp//
        call disp1
        movlw 'R' // data to disp//
```

```
        call disp1
        movlw 'A'// data to disp//
        call disp1
        movlw 'T' // data to disp//
        call disp1
        movlw 'I'// data to disp//
        call disp1
        movlw 'O' // data to disp//
        call disp1

        movlw 'N' // data to disp//
        call disp1
        return

countersetting://counter setting//
            movlw 0x00
            movwf 0x0e//set tmr1l 00//
            movwf 0x0f//tmr1h 00//
            bsf 0x10,1//t1con tmr1cs make it counter//
            bcf 0x10,2//t1con t1sync make it sync//
            bcf 0x10,3//t1con oscillator shut down//
            bcf 0x10,4
            bcf 0x10,5
            bsf 0x87,0//tris_c=for setting pinC,0 as input//
            return

hba2:
        #endasm
        du=0;
        dt=0;

        du=x%10;
        du=du+48;
        dt=x/10;
        dt=dt+48;
        #asm
        return

disp2:
            call ready
```

```
                bsf 0x08,0//rs//
                bcf 0x08,1//rw//
                bsf 0x08,2//enable//
                nop
                nop
                nop
                nop
             #endasm
            output_b(dt);

             #asm
            nop
            nop
            nop
            nop
             bcf 0x08,2
              call ready
                bsf 0x08,0//rs//
                bcf 0x08,1//rw//
                bsf 0x08,2//enable//
                nop
                nop
                nop
                nop
             #endasm
            output_b(du);
              #asm
            nop
            nop
            nop
            nop
             bcf 0x08,2
              return


hba3: #endasm
        du=0;
        dt=0;
        dh=0;
        du=x%10;
```

```
        du=du+48;
        x=x/10;
        dt=x%10;
        dt=dt+48;
        dh=x/10;
        dh=dh+48;
        #asm
        return

disp3:
         call ready

             bsf 0x08,0//rs//
             bcf 0x08,1//rw//
             bsf 0x08,2//enable//
             nop
             nop
             nop
             nop
     #endasm
    output_b(dh);
#asm
    nop
    nop
    nop
    nop
     bcf 0x08,2
        call ready
             bsf 0x08,0//rs//
             bcf 0x08,1//rw//
             bsf 0x08,2//enable//
             nop
             nop
             nop
             nop
     #endasm
    output_b(dt);
         #asm
    nop
    nop
```

```
        nop
        nop
         bcf 0x08,2
            call ready
                bsf 0x08,0//rs//
                bcf 0x08,1//rw//
                bsf 0x08,2//enable//
                nop
                nop
                nop
                nop
         #endasm

        output_b(du);
             #asm
        nop
        nop
        nop
        nop
         bcf 0x08,2
 return

sensedisp:

          movlw 0x80//1stline ctrl=80 to 8f; 2nd line=c0to7f//
          call command
          movlw 'S' // data to disp//
          call disp1
          movlw 'E' // data to disp//
          call disp1
          movlw 'N' // data to disp//
          call disp1
          movlw 'S' // data to disp//
          call disp1
          movlw 'E' // data to disp//
          call disp1
          movlw ' ' // data to disp//
          call disp1
          movlw 'M'// data to disp//
          call disp1
```

```
        movlw 'O' // data to disp//
        call disp1
        movlw 'D'// data to disp//
        call disp1
        movlw 'E' // data to disp//
        call disp1
         return


countonoff:

            movlw 0x00
            movwf 0x0e//set tmr1l 00//

            movwf 0x0f//tmr1h 00//
            bsf 0x10,0//t1con= turn on ctr1//
            #endasm
            delay_ms(249);
            delay_us(999);
            #asm
           bcf 0x10,0//t1con= trn off ctr1//
         #endasm
            count=get_timer1();
            count*=4;
            #asm
            return

high:   movlw 0x01//clr//
        call command
        movlw 0x80//1stline ctrl=80 to 8f; 2nd line=c0to7f//
        call command
        movlw 'T' // data to disp//
        call disp1
        movlw 'E' // data to disp//
        call disp1
        movlw 'M' // data to disp//
        call disp1
        movlw 'P' // data to disp//
        call disp1
        movlw ' ' // data to disp//
```

```
        call disp1
        movlw 'T' // data to disp//
        call disp1
        movlw 'O'// data to disp//
        call disp1
        movlw 'O' // data to disp//
        call disp1
        movlw ' '// data to disp//
        call disp1
        movlw 'H' // data to disp//
        call disp1
        movlw 'I'//1stline ctrl=80 to 8f; 2nd line=c0to7f//
        call disp1
        movlw 'G'//1stline ctrl=80 to 8f; 2nd line=c0to7f//

        call disp1
        movlw 'H'//1stline ctrl=80 to 8f; 2nd line=c0to7f//
        call disp1
        #endasm
        delay_ms(4000);
        #asm
        return

low:    movlw 0x01//clr//
        call command
        movlw 0x80//1stline ctrl=80 to 8f; 2nd line=c0to7f//
        call command
        movlw 'T' // data to disp//
        call disp1
        movlw 'E' // data to disp//
        call disp1
        movlw 'M' // data to disp//
        call disp1
        movlw 'P' // data to disp//
        call disp1
        movlw ' ' // data to disp//
        call disp1
        movlw 'T' // data to disp//
        call disp1
        movlw 'O'// data to disp//
```

```
        call disp1
        movlw 'O' // data to disp//
        call disp1
        movlw ' '// data to disp//
        call disp1
        movlw 'L' // data to disp//
        call disp1
        movlw 'O'//1stline ctrl=80 to 8f; 2nd line=c0to7f//
        call disp1
        movlw 'W'//1stline ctrl=80 to 8f; 2nd line=c0to7f//
        call disp1
        #endasm
        delay_ms(4000);
        #asm
        return

#endasm
}
```

7 Conclusion

This project successfully demonstrates that the capacitance variation due to change in temperature of a solar cell itself can be used to measure the temperature of the solar cell blanket. Its sensitivity has been measured and is within the limits required for its intended space application.

A temperature calculation algorithm has been designed and implemented as code to perform the programming portion of the project. The code for both calibration and non calibration algorithms have been developed, tested and implemented.

The integrated advantages of the temperature measurement system have been listed below

- It is a novel method for measuring temperature in space
- Gives more accurate results than the existing technique in use by eliminating gradient effect
- Uses a solar cell itself as a sensor along with very minimal additional components
- No new technology/ component is necessary for mounting the sensor or instrumentation
- Sensor could be located anywhere and accuracy is still unaffected
- Even during rapid temperature variation, as the satellite enters eclipse, the sensor undergoes exactly by same variation as the solar cells
- The provision for calibration gives it flexibility of use with any solar cells
- Accuracy is to within 2 decimal places of its temperature in degrees

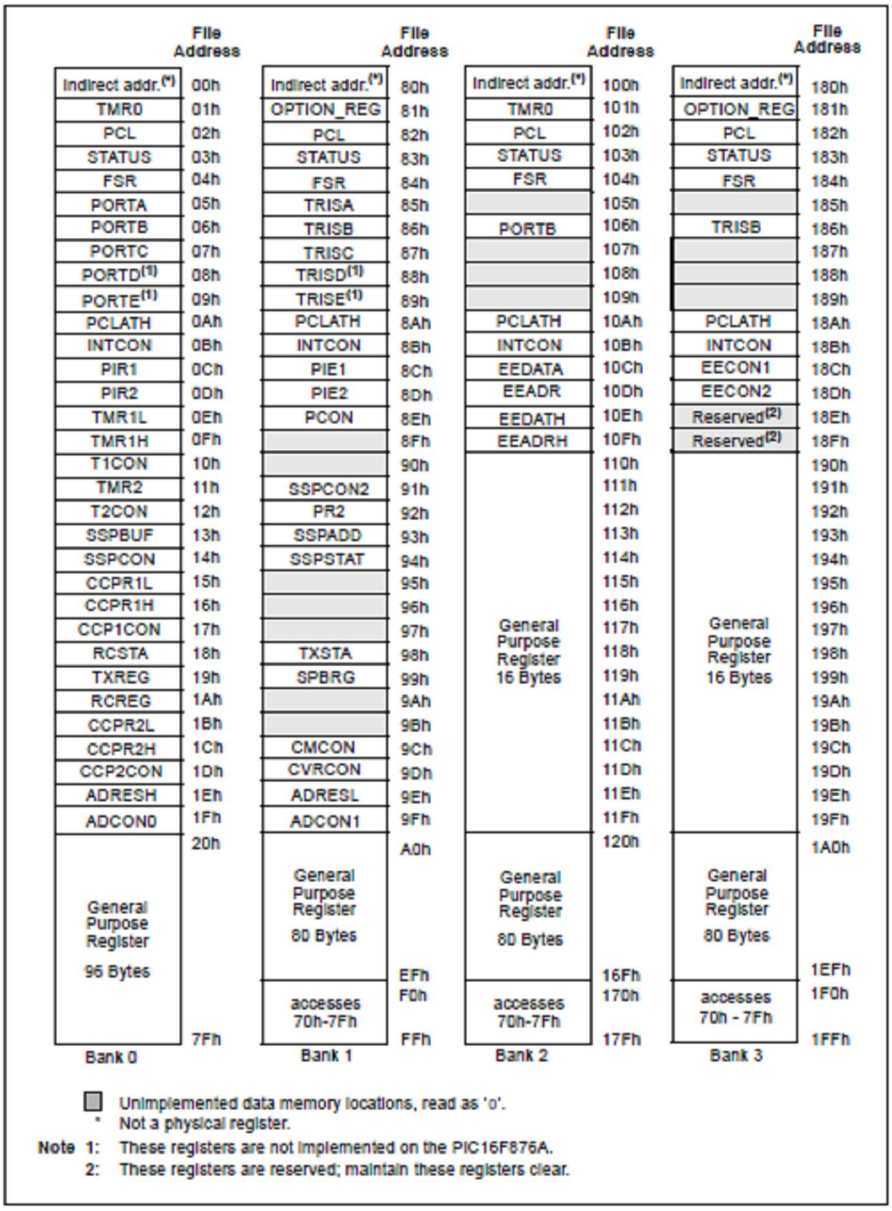
8 References

1. http://www.npl.co.uk/reference/faqs/what-is-a-platinum-resistance-thermometer-(faq-thermal)
2. http://www.npl.co.uk/content-categories/faqs/temp-vs-resistivity
3. Solar cell as a temperature sensor for measuring temperature of solar panels in Satellites (*Journal. of Instrum. Soc. of India* H. Anantha Krishnaa, N.K. Misraa and M.S. Suresh)
4. Error Sources That Effect Platinum Resistance Thermometer Accuracy (John Zwak, Metrology Engineer Burns Engineering, Inc. 2008)
5. http://www.sparkfun.com/commerce/tutorial_info.php?tutorials_id=80 (rs232)
6. Integrated Electronics by Millman- Halkias

9 Appendix

9.1 PIC16F877A- Important registers and other associated documentation

Special function registers



Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3	File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD(1)	08h	TRISD(1)	88h		108h		188h
PORTE(1)	09h	TRISE(1)	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved(2)	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved(2)	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h	General Purpose Register 16 Bytes	117h	General Purpose Register 16 Bytes	197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch	CMCON	9Ch		11Ch		19Ch
CCP2CON	1Dh	CVRCON	9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
General Purpose Register 96 Bytes	20h	General Purpose Register 80 Bytes	A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h
	7Fh		EFh		16Fh		1EFh
		accesses 70h-7Fh	F0h	accesses 70h-7Fh	170h	accesses 70h - 7Fh	1F0h
			FFh		17Fh		1FFh

Unimplemented data memory locations, read as '0'.
* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876A.
2: These registers are reserved; maintain these registers clear.

Timer1 control register

REGISTER 6-1: T1CON: TIMER1 CONTROL REGISTER (ADDRESS 10h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **T1CKPS1:T1CKPS0:** Timer1 Input Clock Prescale Select bits
11 = 1:8 prescale value
10 = 1:4 prescale value
01 = 1:2 prescale value
00 = 1:1 prescale value

bit 3 **T1OSCEN:** Timer1 Oscillator Enable Control bit
1 = Oscillator is enabled
0 = Oscillator is shut-off (the oscillator inverter is turned off to eliminate power drain)

bit 2 **$\overline{\text{T1SYNC}}$:** Timer1 External Clock Input Synchronization Control bit
When TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS:** Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T1CKI (on the rising edge)
0 = Internal clock (Fosc/4)

bit 0 **TMR1ON:** Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Timer1 register pair

6.7 Resetting of Timer1 Register Pair (TMR1H, TMR1L)

TMR1H and TMR1L registers are not reset to 00h on a POR, or any other Reset, except by the CCP1 and CCP2 special event triggers.

T1CON register is reset to 00h on a Power-on Reset, or a Brown-out Reset, which shuts off the timer and leaves a 1:1 prescale. In all other Resets, the register is unaffected.

6.8 Timer1 Prescaler

The prescaler counter is cleared on writes to the TMR1H or TMR1L registers.

TABLE 6-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh,8Bh, 10Bh, 18Bh	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	--00 0000	--uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.
Note 1: Bits PSPIE and PSPIF are reserved on the 28-pin devices; always maintain these bits clear.

Instruction set summary

Mnemonic, Operands		Description	Cycles	14-Bit Opcode				Status Affected	Notes
				MSb			LSb		
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	lfff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add Literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND Literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call Subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to Address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR Literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move Literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from Interrupt	2	00	0000	0000	1001		
RETLW	k	Return with Literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into Standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from Literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR Literal with W	1	11	1010	kkkk	kkkk	Z	

9.2 74123 – Dual retriggerable monostable multivibrator

FAIRCHILD SEMICONDUCTOR TM

March 1998

DM74123
Dual Retriggerable One-Shot with Clear and Complementary Outputs

General Description

The '123 is a dual retriggerable monostable multivibrator capable of generating output pulses from a few nano-seconds to extremely long duration up to 100% duty cycle. Each device has three inputs permitting the choice of either leading-edge or trailing edge triggering. Pin (A) is an active-low transition trigger input and pin (B) is an active-high transition trigger input. A low at the clear (CLR) input terminates the output pulse: which also inhibits triggering. An internal connection from CLR to the input gate makes it possible to trigger the circuit by a positive-going signal on CLR as shown in the truth table.

To obtain the best and trouble free operation from this device please read the operating rules as well as the NSC one–shot application notes carefully and observe recommendations.

Features

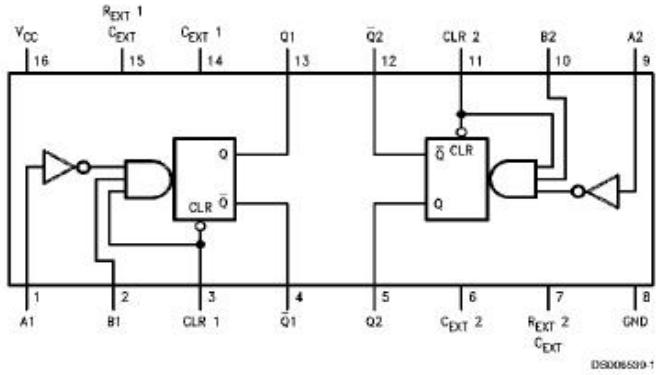
- DC triggered from active-high transition or active-low transition inputs
- Retriggerable to 100% duty cycle
- Direct reset terminates output pulse
- Compensated for V_{CC} and temperature variations
- DTL, TTL compatible
- Input clamp diodes

Functional Description

The basic output pulse width is determined by selection of an external resistor (R_X) and capacitor (C_X). Once triggered, the basic pulse width may be extended by retriggering the gated active-low transition or active-high transition inputs or be reduced by use of the active-low transition clear input. Retriggering to 100% duty cycle is possible by application of an input pulse train whose cycle time is shorter than the output cycle time such that a continuous "HIGH" logic state is maintained at the "Q" output.

Connection Diagram

Dual-In-Line Package



Triggering Truth Table

Inputs			Response
A	B	CLR	
X	X	L	No Trigger
↘	L	X	No Trigger
↘	H	H	Trigger
H	↗	X	No Trigger
L	↗	H	Trigger
L	H	↗	Trigger

H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial

9.3 LM311 – Single comparator

FAIRCHILD
SEMICONDUCTOR®

www.fairchildsemi.com

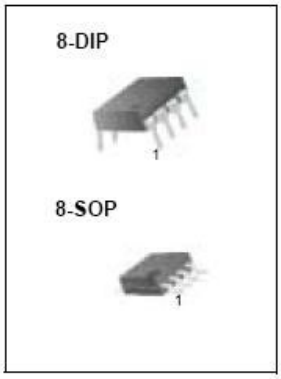
LM311
Single Comparator

Features

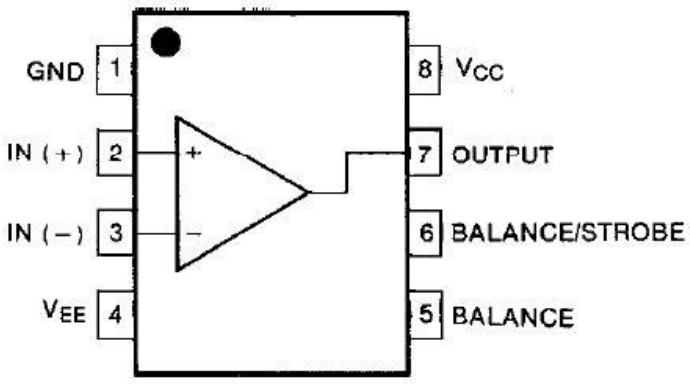
- Low input bias current : 250nA (Max)
- Low input offset current : 50nA (Max)
- Differential Input Voltage : ±30V
- Power supply voltage : single 5.0V supply to ±15V.
- Offset voltage null capability.
- Strobe capability.

Description

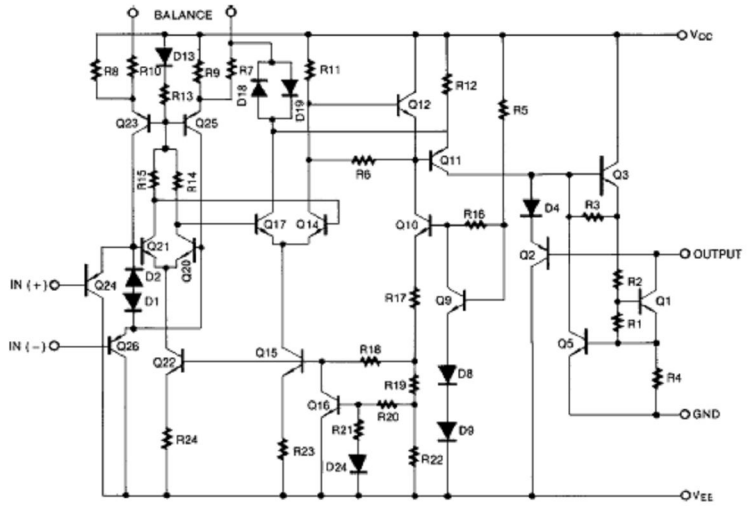
The LM311 series is a monolithic, low input current voltage comparator. The device is also designed to operate from dual or single supply voltage.



Internal Block Diagram



Schematic Diagram



Absolute Maximum Ratings

Parameter	Symbol	Value	Unit
Total Supply Voltage	V_{CC}	36	V
Output to Negative Supply Voltage LM311	V_O - V_{EE}	40	V
Ground to Negative voltage	V_{EE}	-30	V
Differential Input Voltage	$V_{I(DIFF)}$	30	V
Input Voltage	V_I	±15	V
Output Short Circuit Duration	-	10	sec
Power Dissipation	P_D	500	mW
Operating Temperature Range	T_{OPR}	0 ~ +70	°C
Storage Temperature Range	T_{STG}	- 65 ~ +150	°C

9.4 LM317 - 3-Terminal Positive Adjustable Regulator

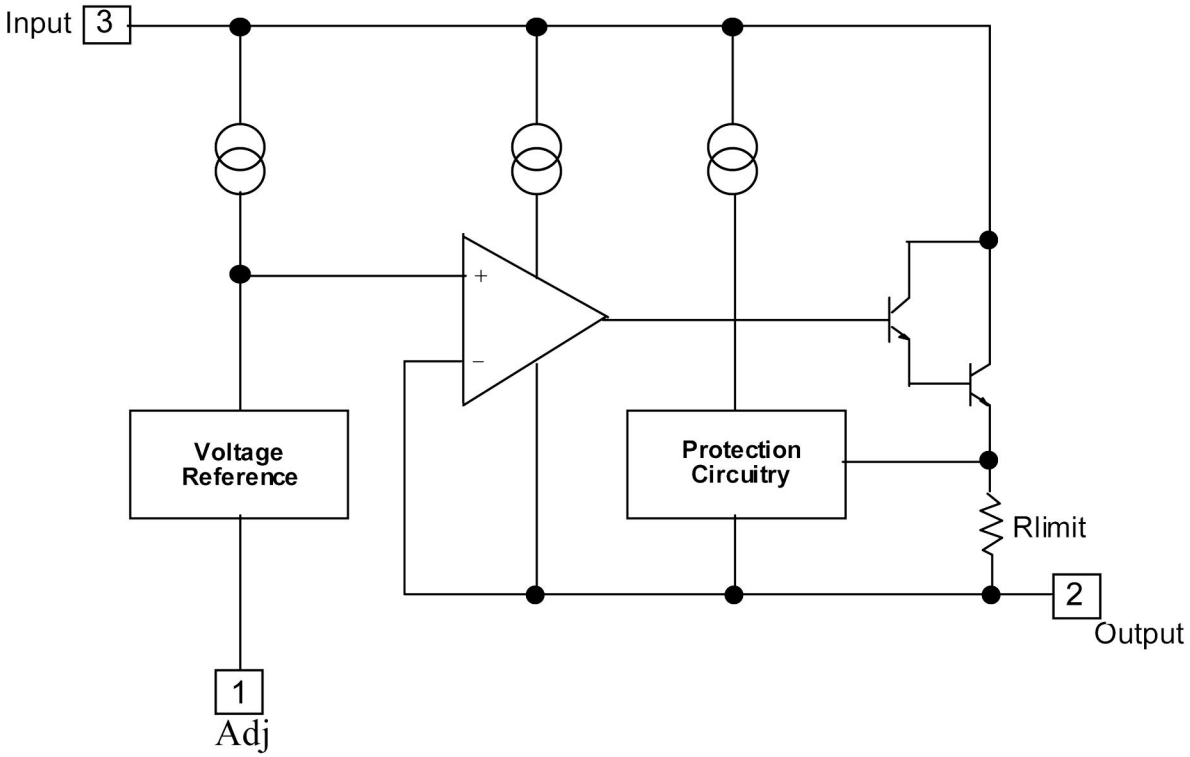
General Description

This monolithic integrated circuit is an adjustable 3-terminal positive voltage regulator designed to supply more than 1.5A of load current with an output voltage adjustable over a 1.2 to 37V. It employs internal current limiting, thermal shut-down and safe area compensation.

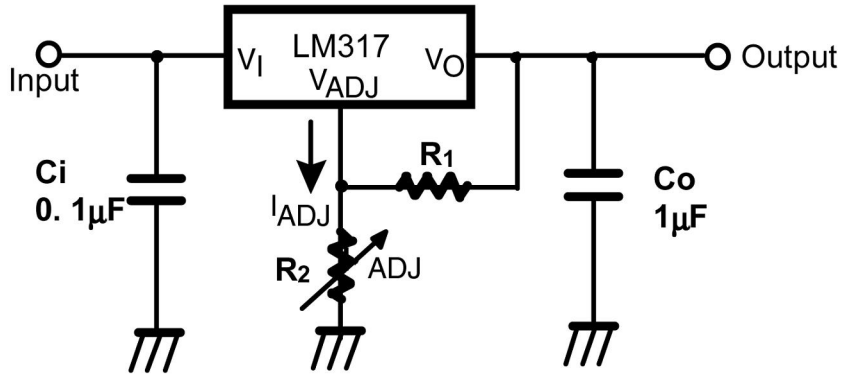
Features

_ Output Current In Excess of 1.5A
_ Output Adjustable Between 1.2V and 37V
_ Internal Thermal Overload Protection
_ Internal Short Circuit Current Limiting
_ Output Transistor Safe Operating Area Compensation
_ TO-220 Package
_ D2 PAK Package

Internal Block Diagram



Typical Application



$$V_O = 1.25V\ (1 + R_2 / R_1) + I_{ADJ}\ R_2$$

9.5 74LS06 – HEX inverter/buffer

Features:

• Converts TTL Voltage Levels to MOS Levels

• High Sink-Current Capability

• Input Clamping Diodes Simplify System Design

• Open-Collector Driver for Indicator Lamps and Relays

• Package Options Include “Small Outline” Packages, Ceramic Chip Carriers, and Standard Plastic and Ceramic 300-mil DIPs

Description

These monolithic hex inverter buffers/drivers feature high-voltage open-collector outputs to interface with high-level circuits (such as MOS), or for driving high-current loads, and are also characterized for use as inverter buffers for driving TTL inputs. The 4LS06 has a rated output voltage of 30 V and the 4LS16 has a rated output voltage of 15 V. The maximum sink current for the SN54LS06 and SN54LS16 is 30 mA and the SN74LS06 and SN74LS16 is 40 mA.These circuits are compatible with most TTL families. Inputs are diode-clamped to minimize transmission-effects, which simplifies design. Typical power dissipation is 175 mW and average propagation delay time is 8 ns. The SN54LS06 and SN54LS16 are characterized over the full military temperature range of –55°C to 125°C.The SN74LS06 and SN74LS16 are characterized for operation from 0°C to 70°C.

schematic (each gate)