REAL TIME WEB BASED MULTI PURPOSE WIRELESS SENSOR NETWORK

A PROJECT REPORT

Submitted by

K.JAGANATH	**71204106015**
V.KATHIRESAN	**71204106025**
M.SARAVANAN	**71204106046**
M.SATHESH KUMAR	**71204106048**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY :: CHENNAI 600025

APRIL 2008

BONAFIDE CERTIFICATE

Certified that this project report **"REAL TIME WEB BASED MULTI PURPOSE WIRELESS SENSOR NETWORK"** is the bonafide work of **"K.JAGANATH, V.KATHIRESAN, M.SARAVANAN, M.SATHESH KUMAR"** who carried out the project work under my supervision.

SIGNATURE

Dr. RAJESWARI MARIAPPAN Ph.D.,

HEAD OF THE DEPARTMENT

Electronics &Communication Engineering,
Kumaraguru College of Technology,
Coimbatore-641006.

SIGNATURE

Mr.G.C.THIYAGARAJAN M.E.,

SUPERVISOR

Senior Lecturer
Electronics &Communication Engineering,
Kumaraguru College of Technology,
Coimbatore-641006.

The candidates with University Register Nos. 71204106015, 71204106025, 71204106046, 71204106048 were examined by us in the project viva-voce examination held on 19.4.08

19/4/08

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We are greatly indebted to our beloved Principal **Dr.Joseph.V.Thanikal, Ph.D.,** Who has been the backbone of all our deeds.

We Profusely thank **Dr. Rajeshwari Mariappan, Ph.D.,** Head of Department, Department of Electronics and Communication Engineering, for lending a help hand in this project.

We are highly grateful to our beloved Project coordinator **Ms. R.Latha, M.E.,,** and Project guide **Mr.G.C.THIYAGARAJAN.,** Senior Lecturer, E.C.E Department for their valuable guidance, timely helps, constant encouragement and advice rendered throughout the project period for the successful completion of the project

We are also grateful to the faculty members of Electronics Department of Electronics and Communication Engineering, Who have helped us in innumerable ways.

We also thank our parents without whom we could not have come so far and friends for their timely help that culminated as good in end.

ABSTRACT

We design a system that allows simultaneous temperature and level monitoring on two different areas in an industry using multiple wireless sensor networks. The distribution of temperature sensor and level sensor senses the temperature and level of a boiler for example. The sensed readings are transmitted to the server through wireless. From the server the real time readings are uploaded to the website. The existence of this wireless sensor network allows for real-time temperature and level monitoring with the temperature and level information being accessed over the Internet. The system we design allows for multiple sensor networks to be integrated and for a user to monitor temperature and level from any where in the world through website. In future this project can be extended by adding more sensors and also can be extended to control the parameters.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

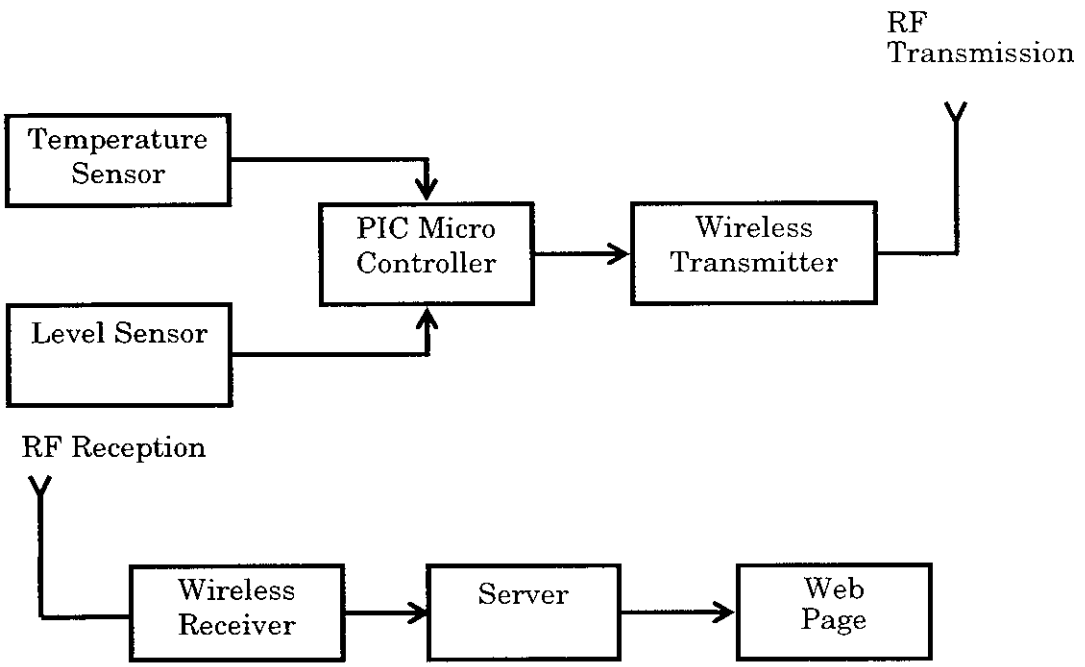
1.1 BRIEF OVERVIEW

In this developing world the man power in the industries are reducing. We are developing a system which reduces the man power. Usually in industries we need a person to note the temperature and level measurements. The person has to be with that boiler and note the readings at certain time interval. In our system the temperature and level readings are automatically sensed and transmitted through wireless and uploaded into the website. So anybody can monitor those readings from anywhere in this world.

We are using negative temperature coefficient resistor. As the temperature increases, resistance will decrease. Accordingly voltage varies. This analog value is send to ADC of PIC. Similarly we are using float for level measurement. As the level changes, accordingly voltage varies. This analog value is send to ADC of PIC.

From port B of PIC the digital readings are sent to encoder and transmitted via wireless. In the receiver section the digital values are received and decoded. Then the readings are sending to server. In the server we can see the output using LabVIEW. From server the readings are uploaded into website. So we can monitor from anywhere in this world.

1.2 BLOCK DIAGRAM



1.3 MODULES

A brief overview of the project modules are explained below for better understanding of block diagram:

- Temperature sensor-senses the atmospheric temperature surrounding the machine.
- Pressure sensor-senses the pressure in the cylinder.
- The analog signal from temperature and level sensor are amplified and given to PIC controller.
- RF Transmitter and Receiver are used to transmit the signal from PIC and receive the same and display it with the help of LabVIEW.
- The received value is uploaded in to a web page using a SQL database table.

CHAPTER 2
WIRELESS SENSOR NETWORKS

2.1 INTRODUCTION

Efficient design and implementation of wireless sensor networks has become a hot area of research in recent years, due to the vast potential of sensor networks to enable applications that connect the physical world to the virtual world. By networking large numbers of tiny sensor nodes, it is possible to obtain data about physical phenomena that was difficult or impossible to obtain in more conventional ways. In the coming years, as advances in micro-fabrication technology allow the cost of manufacturing sensor nodes to continue to drop, increasing deployments of wireless sensor networks are expected, with the networks eventually growing to large numbers of nodes. Potential applications for such large-scale wireless sensor networks exist in a variety of fields, including medical monitoring ,environmental monitoring, surveillance, home security, military operations, and industrial machine monitoring.

2.2 TAXONOMY

As research in sensor networks has grown, so too has the range of applications proposed to make use of this rich source of data. Such diversity of sensor network applications translates to differing requirements from the underlying sensor network. To address these varying needs, many different network models have been proposed, around which protocols for different layers of the network stack have been designed. While there are many ways to classify different sensor network architectures, the following list highlights some fundamental differences in sensor networks that affect protocol design.

• Data sink(s). One of the most important aspects of a sensor network is the nature of the data sink(s). In some situations, the end user(s) may be embedded within the

sensor network or may be less accessible mobile access points that collect data once in a while. This distinction may be important, as efficient distributed data storage techniques may be effective in the latter scenario.

• Sensor mobility. Another classification of sensor networks may be made based on the nature of the sensors being deployed. Typically, it can be assumed that sensors are immobile; however, some recent sensor networks projects such as the ZebraNet project have used mobile sensor nodes. Also, in military operations, additional sensors may be mounted on soldiers or UAVs to interact with a deployed sensor network. The mobility of sensors can influence protocols at the networking layer as well as those for localization services.

• Sensor resources. Sensor nodes may vary greatly in the computing resources available. It is obvious that memory and processing constraints should influence protocol design at nearly every level.

2.3 UNIQUE FEATURES

It should be noted that sensor networks do share some commonalities with general ad hoc networks. Thus, protocol design for sensor networks must account for the properties of ad hoc networks, including the following.

• Lifetime constraints imposed by the limited energy supplies of the nodes in the network.

• Unreliable communication due to the wireless medium.

• Sensor nodes are typically immobile, meaning that the mechanisms used in traditional ad hoc network protocols to deal with mobility may be unnecessary and overweight.

• Since nodes may be deployed in harsh environmental conditions, unexpected node failure may be common.

• Additional services, such as location information, may be required in wireless sensor networks.

• Sensor networks often have a many-to-one traffic pattern, which leads to a "hot spot" problem. Incorporating these unique features of sensor networks into protocol design is important in order to efficiently utilize the limited resources of the network. At the same time, to keep the protocols as light-weight as possible, many designs focus on particular subsets of these criteria for different types of applications. This has led to quite a number of different protocols from the data-link layer up to the transport layer, each with the goal of allowing the network to operate autonomously for as long as possible while maintaining data channels and network processing to provide the application's required quality of service.

2.4 ROUTING PROTOCOLS

Though there are number of protocols, the mostly used protocols are

- Resource-Aware Routing
- Data-Centric Routing
- Geographic Routing

2.4.1 DATA CENTRIC ROUTING PROTOCOLS

Sensor networks are fundamentally different from ad hoc networks in the data they carry. While in ad hoc networks individual data items are important, in sensor networks it is the aggregate data or the information carried in the data rather than the actual data itself that is important. This has led to a new paradigm for networking these types of devices – data-centric routing. In data-centric routing, the end nodes, the sensors themselves, are less important than the data itself. Thus, queries are posed for specific data rather than for data from a particular sensor, and routing is performed using knowledge that it is the aggregate data rather than any individual data item that is important.

2.4.1.1 SENSOR PROTOCOL FOR INFORMATION VIA NEGOTIATION (SPIN)

SPIN is a protocol that was designed to enable data-centric information dissemination in sensor networks. Rather than blindly broadcasting sensor data throughout the network, nodes receiving or generating data first advertise this data through star advertise messages. The advertise messages simply consist of an application-specific meta-data description of the data itself. This meta-data can describe such aspects as the type of data and the location of its origin. Nodes that are interested in this data request the data from the advertise sender through REQ messages. Finally, the data is disseminated to the interested nodes through DATA messages that contain the data. This procedure is illustrated in Figure. The advantage of SPIN over blind flooding or gossiping data dissemination methods is that it avoids three costly problems: implosion, overlap and resource blindness. Implosion occurs in highly connected networks that employ flooding and thus each sensor receives many redundant copies of the data. For large data messages, this wastes considerable energy. In SPIN, on the other hand, short advertise messages will suffer from the implosion problem, but the costly transfer of data messages is greatly reduced. Overlap occurs due to the redundant nature of sensor data. Thus two sensors with some common data will both send their data, causing redundancy in data transmission and thus energy waste. SPIN is able to solve this problem by naming data so that sensors only request the data or parts of data they are interested in receiving. Finally, in SPIN, there are mechanisms whereby a sensor that is running low on energy will not advertise its data in order to save its dwindling energy resources. Thus SPIN solves the resource blindness problem by having sensors make decisions based on the current level of available resources

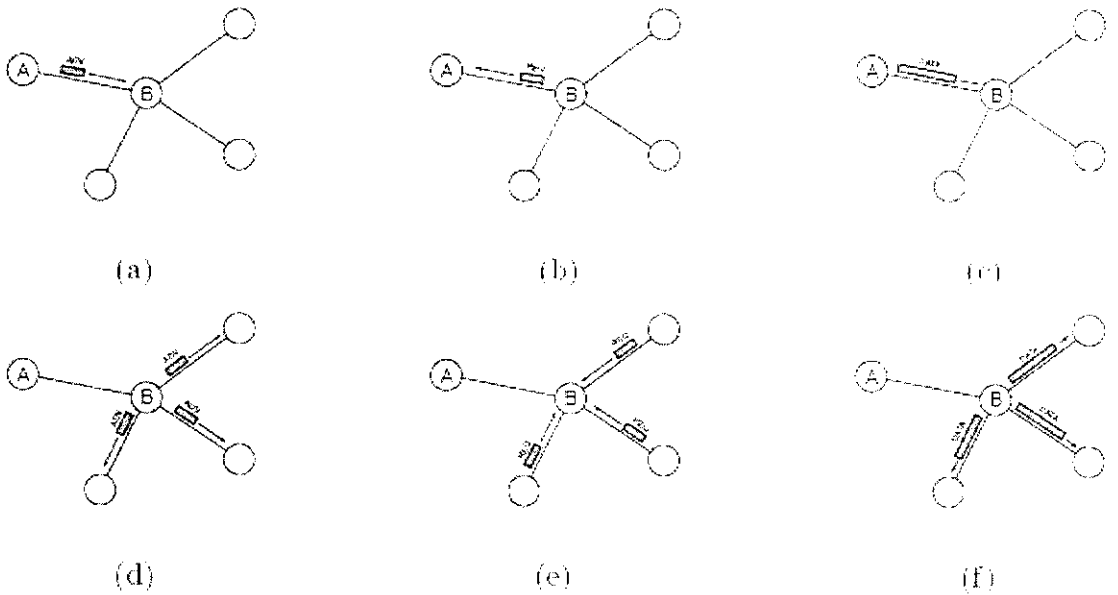


Fig 2.4.1.1 Nodes advertise their data with advertise messages (a). Any node interested in receiving the data replies with a REQ message (b), to which the source node replies with the transmission of the actual data (c). The receiving node then advertises this new data (d) and the processes continues (e,f).

CHAPTER 3
TEMPERATURE SENSOR

3.1 INTRODUCTION

A Thermistor is a type of resistor used to measure temperature changes, relying on the change in its resistance with changing temperature. Thermistor is a combination of the words thermal and resistor. The Thermistor was first invented by Samuel Ruben in 1930, and has U.S. Patent #2,021,491.

If we assume that the relationship between resistance and temperature is linear (i.e. we make a first-order approximation), then we can say that:

$$\Delta R = k\Delta T$$

Where

ΔR = change in resistance

ΔT = change in temperature

k = first-order temperature coefficient of resistance

Thermistors can be classified into two types depending on the sign of k. If k is positive, the resistance increases with increasing temperature, and the device is called a positive temperature coefficient (**PTC**) thermistor, Posistor. If k is negative, the resistance decreases with increasing temperature, and the device is called a negative temperature coefficient (NTC) thermistor. Resistors that are not thermistors are designed to have the smallest possible k, so that their resistance remains almost constant over a wide temperature range.

3.2 CIRCUIT DIAGRAM

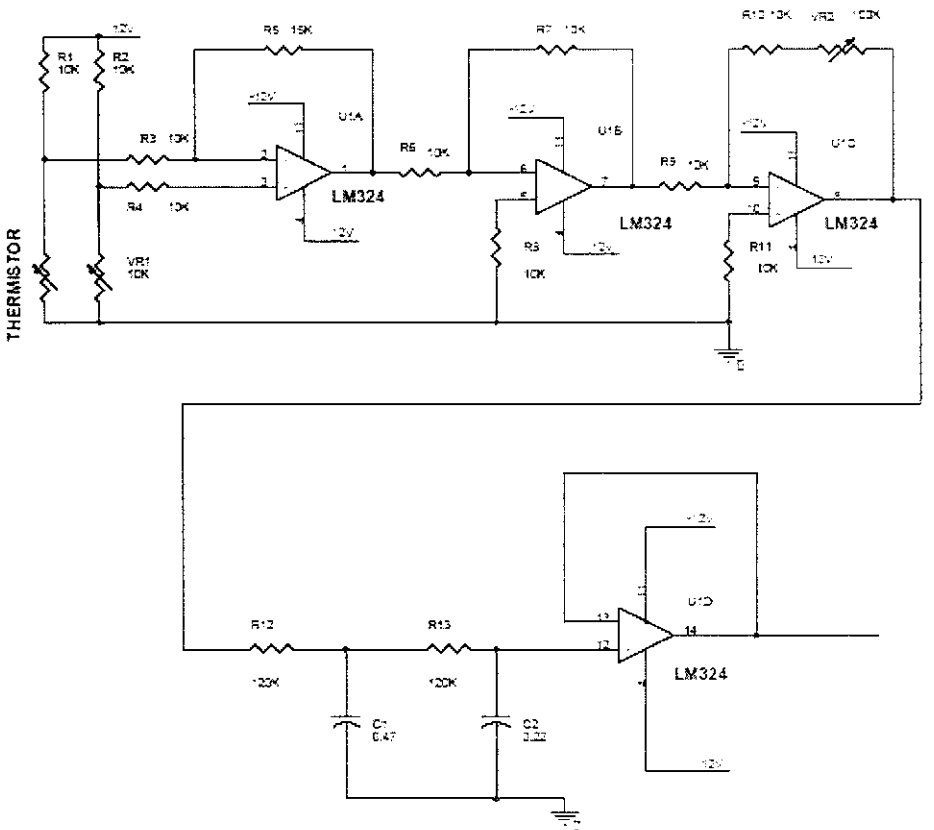


Fig 3.2.1 Temperature Sensor Circuit

3.3 CIRCUIT DESCRIPTION

In this circuit the thermistor is used to measure the temperature. Thermistor is nothing but temperature sensitive resistor. There are two type of thermistor available such as positive temperature co-efficient and negative temperature co-efficient. Here we are using negative temperature co-efficient in which the resistance value is decreased when the temperature is increased.

Here the thermistor is connected with resister bridge network. The bridge terminals are connected to inverting and non-inverting input terminals of comparator. The comparator is constructed by LM 324 operational amplifier.

The LM 324 consist of four independent, high gains, internally frequency compensated operational amplifier which were designed specifically to operate from a single power supply over a wide voltage range.

The first stage is a comparator in which the variable voltage due to thermistor is given to inverting input terminal and reference voltage is given to non-inverting input terminal.

Initially the reference voltage is set to room temperature level so the output of the comparator is zero. When the temperature is increased above the room temperature level, the thermistor resistance is decreased so variable voltage is given to comparator. Now the comparator delivered the error voltage at the output. Then the error voltage is given to next stage of preamplifier. Here the input error voltage is amplified then the amplified voltage is given to next stage of gain amplifier. In this amplifier the variable resistor is connected as feedback resistor. The feedback resistor is adjusted to get desired gain. Then the AC components in the output are filtered with the help of capacitors. Then output voltage is given to final stage of DC voltage follower through this the output voltage is given to ADC or other circuit.

CHAPTER 4
LEVEL MEASUREMENT USING FLOAT

4.1 INTRODUCTION

Level sensors are used to detect liquid level. The liquid to be measured can be inside a container or can be in its natural form (e.g. a river or a lake). The level measurement can be either continuous or point values. Continuous level sensors measure level within a specified range and are used to know the exact amount of liquid in a certain place and Point level sensors only measures a specific level, generally this is used to detect high level alarms or low level alarms.

There are many physical and application variables that affect the selection of the optimal level monitoring solution for industrial and / or commercial processes. The selection criteria include the physical: state (liquid, solid or slurry), temperature, pressure or vacuum, chemistry, dielectric constant of medium, density or specific gravity of medium, agitation, acoustical or electrical noise, vibration, mechanical shock, tank or bin size and shape; and the application constraints: price, accuracy, appearance, response rate, ease of calibration or programming, physical size and mounting of the instrument, monitoring or control of continuous or discrete (point) levels.

4.2 CIRCUIT DIAGRAM

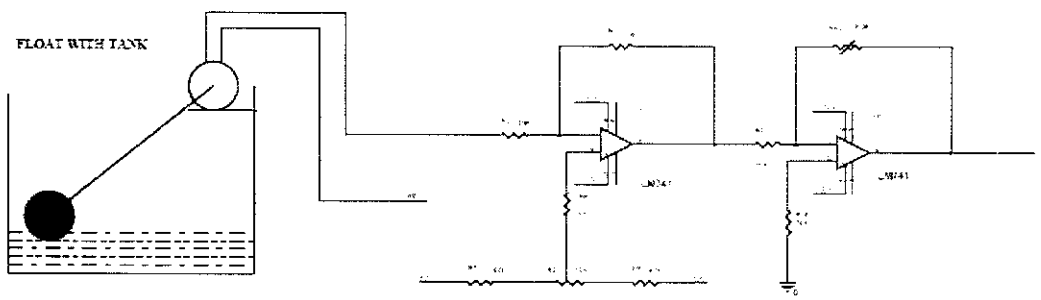


Fig 4.2.1 Level Sensor using Float

Float is the one type of transducer which is used to measure the liquid level in the tank.

4.3 CIRCUIT DESCRIPTION

The float changes the resistance value depending on the water level. This change is resistance is converted into corresponding voltage signal which is given to inverting input terminal of the comparator. The reference voltage is given to non inverting input terminal.

The comparator is constructed by the operational amplifier LM 741. The comparator compares with reference water level and delivered the error voltage at the output terminal. Then the error voltage is given to next stage of gain amplifier which is constructed by another operational amplifier LM 741. In the gain amplifier the variable resistor is connected in the feedback path, by adjusting the resistor we can get the desired gain. Then the final voltage is given to ADC for convert the analog signal to digital signal. Then the corresponding digital signal is given to microcontroller in order to find the water level in the tank.

CHAPTER 5
PIC MICRO CONTROLLER

5.1 INTRODUCTION

The microcontroller that has been used for this project is from PIC series. PIC microcontroller is the first RISC based microcontroller fabricated in CMOS (complementary metal oxide semiconductor) that uses separate bus for instruction and data allowing simultaneous access of program and data memory. The main advantage of CMOS and RISC combination is low power consumption resulting in a very small chip size with a small pin count. The main advantage of CMOS is that it has immunity to noise than other fabrication techniques.

Various microcontrollers offer different kinds of memories. EEPROM, EPROM, FLASH etc. are some of the memories of which FLASH is the most recently developed. Technology that is used in pic16F877 is flash technology, so that data is retained even when the power is switched off. Easy Programming and Erasing are other features of PIC 16F877.

5.2 WHY PIC MICRO CONTOLLER?

- Inbuilt ADC
- RISC (Reduced Instruction Set Code)
- EPROM and Flash memory
- Watchdog timer
- Brown – out reset

5.3 CORE FEATURES

- High-performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of Flash Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM data memory
- Pin out compatible to the PIC16C73/74/76/77
- Interrupt capability (up to 14 internal/external
- Eight level deep hardware stack
- Direct, indirect, and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC Oscillator for reliable operation
- Programmable code-protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low-power, high-speed CMOS EPROM/EEPROM technology
- Fully static design
- In-Circuit Serial Programming (ICSP) via two pins
- Only single 5V source needed for programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.5V to 5.5V

- High Sink/Source Current: 25 mA
- Commercial and Industrial temperature ranges
- Low-power consumption:

  - <2mA typical @ 5V, 4 MHz
  - 20μA typical @ 3V, 32 kHz
  - <1μA typical standby current

5.4 I/O PORTS

Some pins for these I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

5.4.1 PORT A AND THE TRISA REGISTER

PORT A is a 6-bit wide bi-directional port. The corresponding data direction register is TRISA. Setting a TRISA bit (=1) will make the corresponding PORT A pin an input, i.e., put the corresponding output driver in a Hi-impedance mode. Clearing a TRISA bit (=0) will make the corresponding PORT A pin an output, i.e., put the contents of the output latch on the selected pin.

In our project PORT A is used as a input port to receive the signal from Temperature and Level sensors. RA0 pin in PORT A is used for receiving analog signal from temperature sensor and RA1 pin to receive level sensor signal.

5.4.2 PORT B AND TRISB REGISTER

PORT B is an 8-bit wide bi-directional port. The corresponding data direction register is TRISB. Setting a TRISB bit (=1) will make the corresponding PORT B pin an input, i.e., put the corresponding output driver in a hi-impedance mode. Clearing a TRISB bit (=0) will make the corresponding PORT B pin an

output, i.e., put the contents of the output latch on the selected pin. Three pins of PORTB are multiplexed with the Low Voltage Programming function; RB3/PGM, RB6/PGC and RB7/PGD. The alternate functions of these pins are described in the Special Features Section. Each of the PORT B pins has a weak internal pull-up. A single control bit can turn on all the pull-ups.

This is performed by clearing bit RBPU (OPTION_REG<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

PORT B is used as a output port for sending digital data for transmission to the encoder. The data is transmitted in parallel through all 8 pins of PORT B to encoder part.

5.4.3 PORT C AND THE TRISC REGISTER

PORT C is an 8-bit wide bi-directional port. The corresponding data direction register is TRISC. Setting a TRISC bit (=1) will make the corresponding PORT C pin an input, i.e., put the corresponding output driver in a hi-impedance mode. Clearing a TRISC bit (=0) will make the corresponding PORT C pin an output, i.e., put the contents of the output latch on the selected pin. PORT C is multiplexed with several peripheral functions. PORT C pins have Schmitt Trigger input buffers.

5.4.4 PORT D AND TRISD REGISTERS

PORT D is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output. PORT D can be configured as an 8-bit wide microprocessor Port (parallel slave port) by setting control bit PSPMODE (TRISE<4>). In this mode, the input buffers are TTL.

PORT D is used to connect the microcontroller to LCD display. Here PORT D is assigned as an output port.

5.4.5 PORT E AND TRISE REGISTER

PORT E has three pins RE0/RD/AN5, RE1/WR/AN6 and RE2/CS/AN7, which are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers.

The PORT E pins become control inputs for the microprocessor port when bit PSPMODE (TRISE<4>) is set. In this mode, the user must make sure that the TRISE<2:0> bits are set (pins are configured as digital inputs). Ensure ADCON1 is configured for digital I/O. In this mode the input buffers are TTL.

PORT E pins are multiplexed with analog inputs. When selected as an analog input, these pins will read as '0's. TRISE controls the direction of the RE pins, even when they are being used as analog inputs. The user must make sure to keep the pins configured as inputs when using them as analog inputs.

PORT E is also used as an input port if the numbers of analog inputs are more than 5, since PORT A can be used to access only 5 analog signals.

5.5 ANALOG TO DIGITAL CONVERTER (ADC)

There are two types of analog to digital converter is present in this IC. We use 10-bit ADC. The ADC module can have up to eight analog inputs for a device. The analog input charges a sample and hold capacitor. The output of sample and hold capacitor is the input into the converter. The converter then generates a digital result of this analog level via successive approximation. The A/D conversion of the analog input signal results in a Corresponding 10-bit digital number. The A/D module has high and low voltage reference input that is software selectable to some combination of VDD, VSS, and RA2 or RA3.The A/D module has four registers.

These registers are

A/D RESULT HIGH REGISTER (ADRESH)

A/D RESULT LOW REGISTER (ADRESL)

A/D CONTROL REGISTER 0 (ADCON0)

A/D CONTROL REGISTER 1 (ADCON1)

The ADCON0 register, shown in Register 11-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 11-2, configures the functions of the port pins. The port pins can be configured as analog inputs (RA3 can also be the voltage reference) or as digital I/O.

Refer APPENDIX B for more information about micro controller and its ADC conversion along with ADCON registers.

5.6 LIQUID CRYSTAL DISPLAY (LCD)

Liquid crystal displays (LCDs) have materials, which combine the properties of both liquids and crystals. Rather than having a melting point, they have a temperature range within which the molecules are almost as mobile as they would be in a liquid, but are grouped together in an ordered form similar to a crystal. An LCD consists of two glass panels, with the liquid crystal material sand witched in between them. The inner surface of the glass plates are coated with transparent electrodes which define the character, symbols or patterns to be displayed polymeric layers are present in between the electrodes and the liquid crystal, which makes the liquid crystal molecules to maintain a defined orientation angle.

One each polarizes are pasted outside the two glass panels. These polarizes would rotate the light rays passing through them to a definite angle, in a particular direction. When the LCD is in the off state, light rays are rotated by the two polarizes and the liquid crystal, such that the light rays come out of the LCD

without any orientation, and hence the LCD appears transparent. When sufficient voltage is applied to the electrodes, the liquid crystal molecules would be aligned in a specific direction. The light rays passing through the LCD would be rotated by the polarizes, which would result in activating / highlighting the desired characters. The LCD's are lightweight with only a few millimeters thickness. Since the LCD's consume less power, they are compatible with low power electronic circuits, and can be powered for long durations.

The LCD does not generate light and so light is needed to read the display. By using backlighting, reading is possible in the dark. The LCD's have long life and a wide operating temperature range. Changing the display size or the layout size is relatively simple which makes the LCD's more customers friendly. The LCDs used exclusively in watches, calculators and measuring instruments are the simple seven-segment displays, having a limited amount of numeric data. The recent advances in technology have resulted in better legibility, more information displaying capability and a wider temperature range. These have resulted in the LCDs being extensively used in telecommunications and entertainment electronics. The LCDs have even started replacing the cathode ray tubes (CRTs) used for the display of text and graphics, and also in small TV applications.

Crystalonics dot–matrix (alphanumeric) liquid crystal displays are available in TN, STN types, with or without backlight. The use of C-MOS LCD controller and driver ICs result in low power consumption. These modules can be interfaced with a 4-bit or 8-bit microprocessor /Micro controller.

- The built-in controller IC has the following features:
- Correspond to high speed MPU interface (2MHz)
- 80 x 8 bit display RAM (80 Characters max)

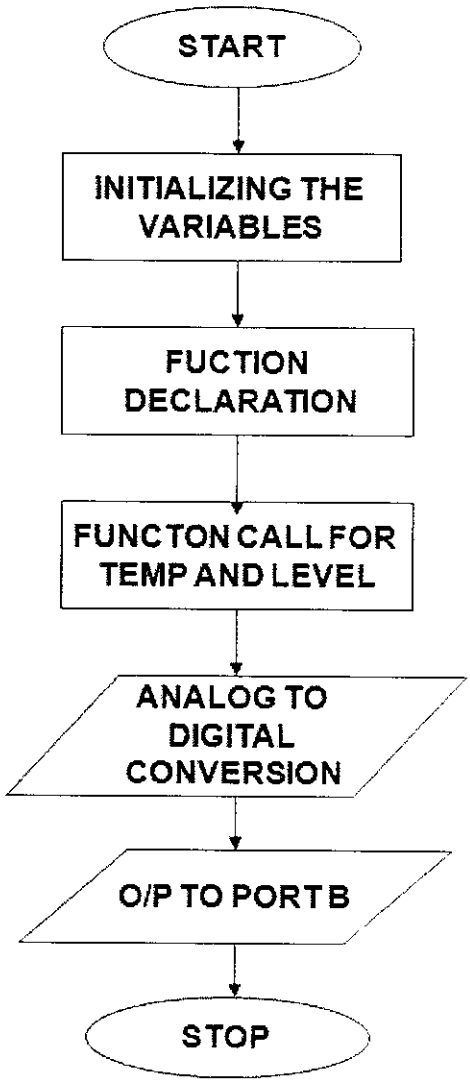
- 9,920-bit character generator ROM for a total of 240 character fonts. 208 character fonts (5 x 8 dots) 32 character fonts (5 x 10 dots)
- 64 x 8 bit character generator RAM 8 character generator RAM 8 character fonts (5 x 8 dots) 4 characters fonts (5 x 10 dots)
- Programmable duty cycles
- 1/8 – for one line of 5 x 8 dots with cursor
- 1/11 – for one line of 5 x 10 dots with cursor
- 1/16 – for one line of 5 x 8 dots with cursor
- Wide range of instruction functions display clear, cursor home, display on/off, cursor on/off, display character blink, cursor shift, display shift.
- Automatic reset circuit, which initializes the controller / driver ICs after power on.

5.7 MICROCONTROLLER

5.7.1ALGORITHM

- All the functions and variables are declared initially.
- PORT A is assigned as input port and PORT B as output port.
- The analog signal from temperature sensor is received at channel 0 of port A.
- The conversion of analog signal to digital is done with control registers of A/D module (ADCON0, ADCON1).
- The analog signal from level sensor is received at channel 1 of port A.
- The reference bits are included in between the data to be sent.

5.7.2 FLOWCHART



CHAPTER 6

TRANSMISSION AND RECEPTION

6.1 RF TRANSMITTER

6.1.1 CIRCUIT DIAGRAM

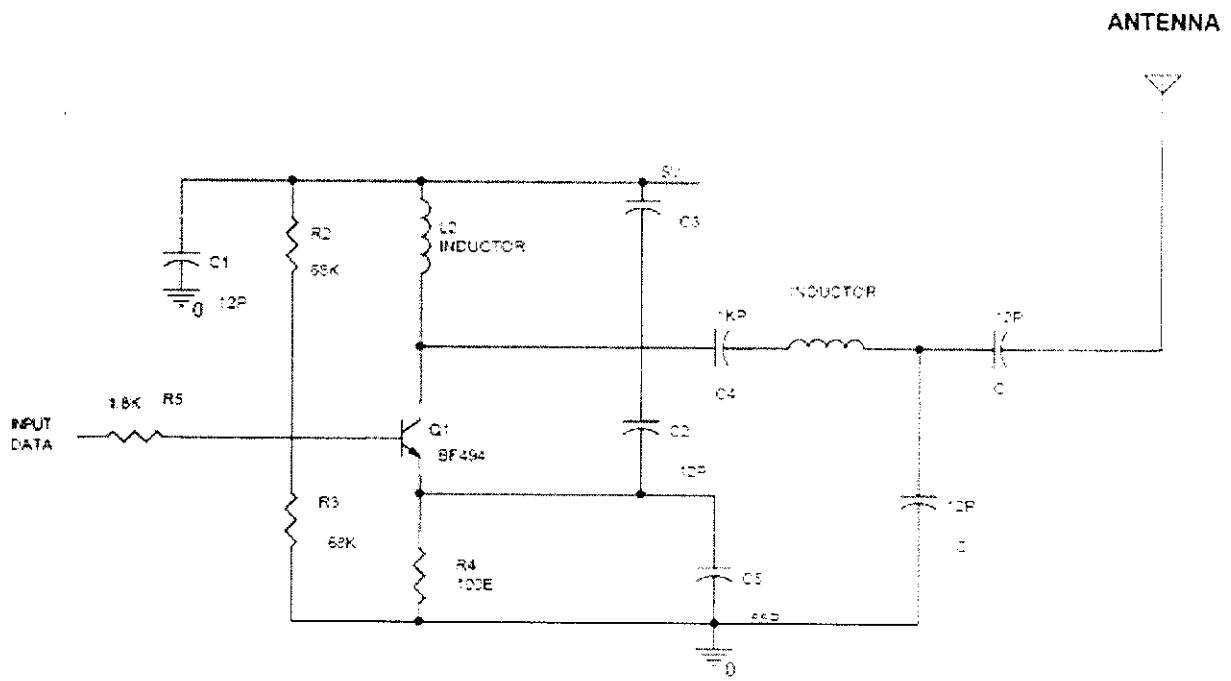


Fig 6.1.1.1 RF Transmitter Circuit

6.1.2 DESCRIPTION

Whenever the high output pulse is given to base of the transistor BF 494, the transistor is conducting so tank circuit is oscillated. The tank circuit is consists of L2 and C4 generating 433 MHz carrier signal. Then the modulated signal is given LC filter section. After the filtration the RF modulated signal is transmitted through antenna.

6.1.3 SPECIFICATIONS

- Frequency Range: 433.92MHz
- Modulation Mode: ASK
- Data Rate: 8Kbps
- Supply Voltage:5V
- Voltage: 5V; Current:8.4mA
- Power Supply and All Input /Output Pins: -0.3 to 12.0 V
- Non-Operating Case Temperature: -10 to 85 centigrade
- Soldering Temperature: 230 centigrade(10 Seconds)
- High Sensitivity Passive Design

Further details about Electrical Characteristic of RF Transmitter and Absolute Maximum Ratings in APPENDIX C

6.2 RF RECEIVER

6.2.1 CIRCUIT DIAGRAM

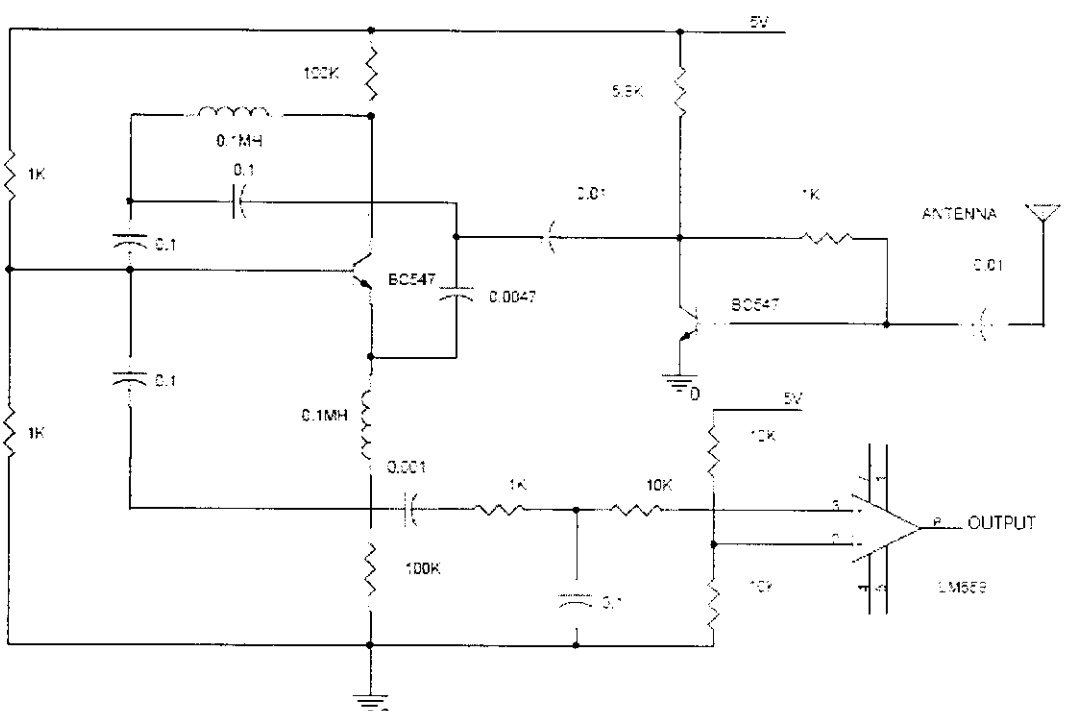


Fig 6.2.1.1 RF Receiver Circuit

6.2.2 DESCRIPTION

The RF receiver is used to receive the encoded data which is transmitted by the RF transmitter. Then the received data is given to transistor which acts as amplifier. Then the amplified signal is given to carrier demodulator section in which transistor Q1 is turn on and turn off conducting depends on the signal. Due to this the capacitor C14 is charged and discharged so carrier signal is removed and saw tooth signal is appears across the capacitor. Then this saw tooth signal is given to comparator. The comparator circuit is constructed by LM558. The comparator is used to convert the saw tooth signal to exact square pulse. Then the encoded signal is given to decoder in order to get the decoded original signal.

6.2.3 SPECIFICATIONS

- Frequency Range: 433.92MHz
- Modulation Mode: ASK
- Data Rate: 4800bps
- High Sensitivity Passive Design
- Supply Voltage:5V
- Channel spacing: +/- 500KHz
- Sensitivity: -106 dBm

Further details about DC Characteristics and Electrical Characteristics of RF Receiver.

6.3 ENCODER

6.3.1 DIAGRAM

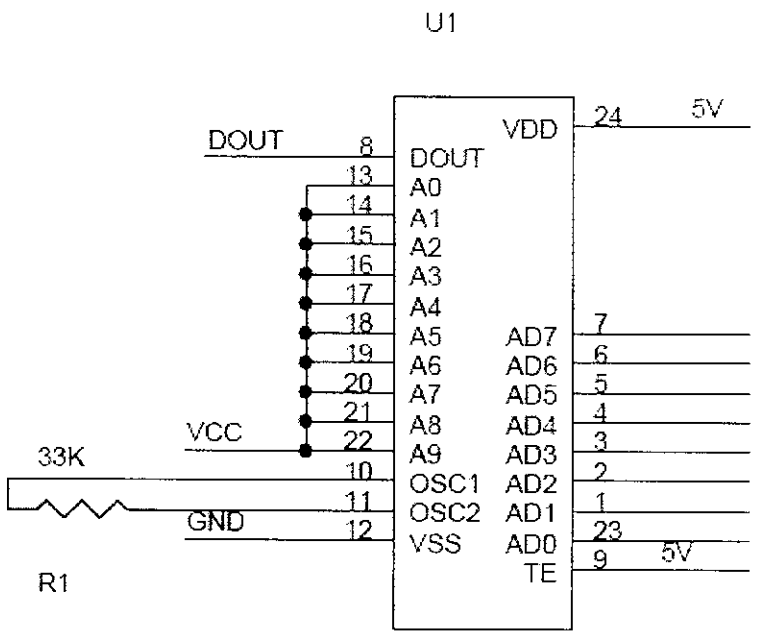


Fig 6.3.1.1 HT640

6.3.2 DESCRIPTION

In this circuit HT 640 is used as encoder. The 3^{18} encoders are a series of CMOS LSIs for remote control system application. They are capable of encoding 18 bits of information which consists of N address bit and 18-N data bits. Each address/data input is externally trinary programmable if bonded out. It is otherwise set floating internally. Various packages of the 3^{18} encoders offer flexible combination of programmable address/data is transmitted together with the header bits via an RF or an infrared transmission medium upon receipt of a trigger signal. The capability to select a TE trigger type further enhances the application flexibility of the 3^{18} series of encoders.

In this circuit the input signal to be encoded is given to AD7-AD0 input pins of encoder. The encoder output address pins are shorted so the output encoded signal is the combination of (A0-A9) address signal and (D0-D7) data signal. The output encoded signal is taken from 8^{th} which is connected to RF transmitter section.

6.4 DECODER

6.4.1 DIAGRAM

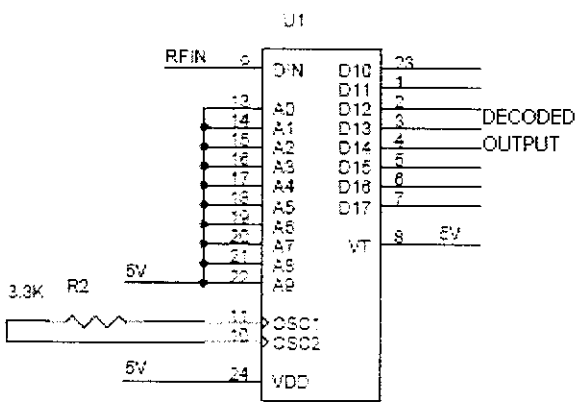


Fig 6.4.1.1 HT648

6.4.2 DESCRIPTION

In this circuit HT648 is used as decoder. The 3^{18} decoder are a series of CMOS LSIs for remote control system application. They are paired with 3^{18} series of encoders. For proper operation a pair of encoder/decoder pair with the same number of address and data format should be selected.

The 3^{18} series of decoder receives serial address and data from that series of encoders that are transmitted by a carrier using an RF or an IR transmission medium. It then compares the serial input data twice continuously with its local address. If no errors or unmatched codes are encountered, the input data codes are

decoded and then transferred to the output pins. The VT pin also goes high to indicate a valid transmission.

The 3^{18} decoders are capable of decoding 18 bits of information that consists of N bits of address and 18-N bits of data. To meet various applications they are arranged to provide a number of data pins whose range is from 0 to 8 and an address pin whose range is from 8 to 18. In addition, the 3^{18} decoders provide various combinations of address/ data numbering different package.

In this circuit the received encoded signal is 9th pin of the decoder. Now the decoder separate the address (A0-A9) and data signal (D0-D7). Then the output data signal is given to microcontroller or any other interfacing device.

CHAPTER 7
LABVIEW

7.1 INTRODUCTION TO LABVIEW

LabVIEW is a graphical programming language that uses icons instead of lines of text to create applications. In contrast to text-based programming languages, where instructions determine program execution, LabVIEW uses dataflow programming, where the flow of data determines execution. In LabVIEW, we build a user interface by using a set of tools and objects. The user interface is known as the front panel. We then add code using graphical representations of functions to control the front panel objects. The block diagram contains this code. In some ways, the block diagram resembles a flowchart.

LabVIEW programs are called virtual instruments, or VIs, because their appearance and operation imitate physical instruments, such as oscilloscopes and multimeters. Every VI uses functions that manipulate input from the user interface or other sources and display that information or move it to other files or other computers.

A VI contains the following three components:

- **Front panel**—Serves as the user interface.
- **Block diagram**—contains the graphical source code that defines the functionality of the VI.
- **Icon and connector pane**—Identifies the VI so that you can use the VI in another VI. A VI within another VI is called a subVI. A subVI corresponds to a subroutine in text-based programming languages.

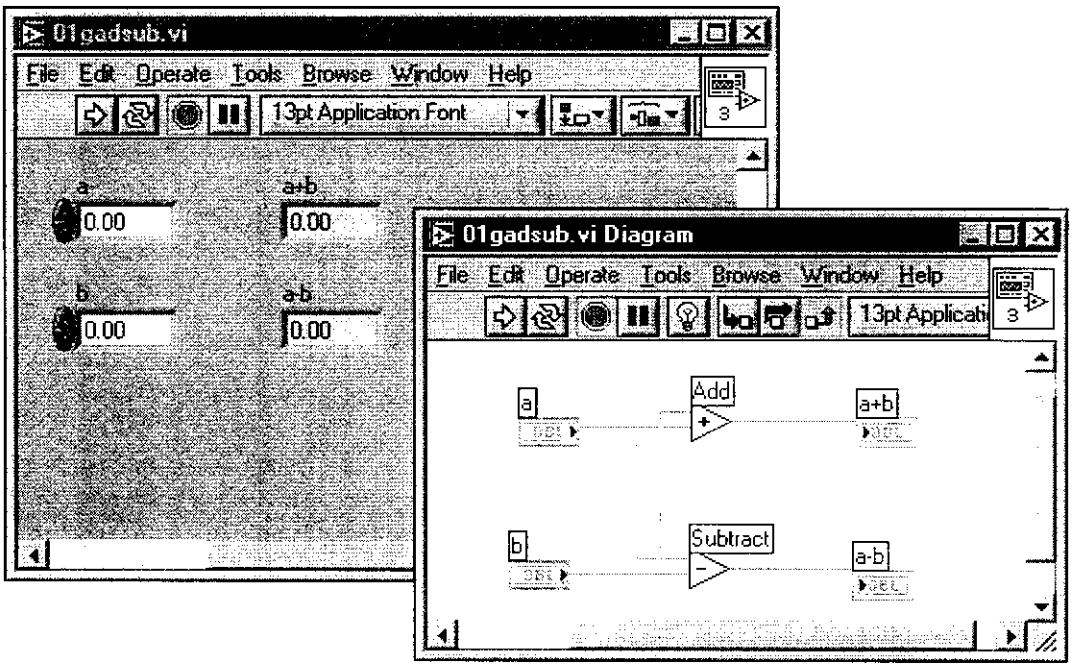


Fig 7.1.1 Example of a Block Diagram and Corresponding Front Panel

7.2 BUILDING THE FRONT PANEL

The front panel is the user interface of a VI. Generally, you design the front panel first, and then design the block diagram to perform tasks on the inputs and outputs we create on the front panel.

We build the front panel with controls and indicators, which are the interactive input and output terminals of the VI, respectively. Controls are knobs, push buttons, dials, and other input devices. Indicators are graphs, LEDs, and other displays. Controls simulate instrument input devices and supply data to the block diagram of the VI. Indicators simulate instrument output devices and display data the block diagram acquires or generates.

Select **Window»Show Controls Palette** to display the **Controls** palette, then select controls and indicators from the **Controls** palette and place them on the front panel.

7.3 BUILDING THE BLOCK DIAGRAM

After we build the front panel, we add code using graphical representations of functions to control the front panel objects. The block diagram contains this graphical source code.

Relationship Between Front Panel Objects And Block Diagram Terminals

Front panel objects appear as terminals on the block diagram. Double-click a block diagram terminal to highlight the corresponding control or indicator on the front panel. Terminals are entry and exit ports that exchange information between the front panel and block diagram. Data we enter into the front panel controls enter the block diagram through the control terminals. When the VI finishes running, the output data flow to the indicator terminals, where they exit the block diagram, re enter the front panel, and appear in front panel indicators.

7.4 FUNCTIONS OVERVIEW

7.4.1Numeric Functions

Use the Numeric functions located on the **Functions»Numeric** palette to create and perform arithmetic, trigonometric, logarithmic, and complex mathematical operations on numbers and to convert numbers from one data type to another.

7.4.2 Boolean Functions

Use the Boolean functions located on the **Functions»Boolean** palette to perform logical operations on single Boolean values or arrays of Boolean values, such as the following tasks:

- Change a TRUE value to a FALSE value and vice versa.
- Determine which Boolean value to return if you receive two or more Boolean values.
- Convert a Boolean value to a number (either 1 or 0).

7.4.3 String Functions

Use the String functions located on the **Functions»String** palette to perform the following tasks:

- Concatenate two or more strings.
- Extract a subset of strings from a string.
- Search for and replace characters or subsets of strings in a string.
- Convert data into strings.
- Format a string for use in a word processing or spreadsheet application.

7.4.4 Array Functions

Use the Array functions located on the **Functions»Array** palette to create and manipulate arrays, such as the following tasks:

- Extract individual data elements from an array.
- Add individual data elements to an array.
- Split arrays.

7.4.5 Cluster Functions

Use the Cluster functions located on the **Functions»Cluster** palette to create and manipulate clusters, such as the following tasks:

- Extract individual data elements from a cluster.
- Add individual data elements to a cluster.
- Break a cluster out into its individual data elements.

7.4.6 Comparison Functions

Use the Comparison functions located on the **Functions»Comparison** palette to compare Boolean values, strings, numeric, arrays, and clusters.

7.4.7 Time and Dialog Functions

Use the Time and Dialog functions located on the **Functions»Time & Dialog** palette to perform the following tasks:

- Manipulate the speed at which an operation executes.
- Retrieve time and date information from your computer clock.
- Create dialog boxes to prompt users with instructions.

7.4.8 File I/O Functions

Use the File I/O functions located on the **Functions»File I/O** palette to perform the following tasks:

- Open and close files.
- Read from and write to files.
- Create directories and files you specify in the path control.
- Retrieve directory information.
- Write strings, numbers, arrays, and clusters to files.

7.4.9 Waveform Functions

Use theWaveform functions located on the **Functions»Waveform** palette to perform the following tasks:

- Build waveforms that include the waveform values, channel information, and timing information.
- Extract individual data elements from a waveform.
- Edit individual data elements of a waveform.

7.5 SAVING VIs

We can save VIs as individual files or we can group several VIs together and save them in a VI library. VI library files end with the extension .llb. National

Instruments recommends that we save VIs as individual files, organized in directories, especially if multiple developers are working on the same project.

7.5.1 Advantages of Saving VIs as Individual Files

The following list describes reasons to save VIs as individual files:

• We can use the file system to manage the individual files.

• We can use subdirectories.

• We can store VIs and controls in individual files more robustly than we can store our entire project in the same file.

• We can use the Professional Development System built-in source code control tools or third-party source code control tools.

7.5.2 Advantages of Saving VIs as Libraries

The following list describes reasons to save VIs as libraries:

• We can use up to 255 characters to name our files.

(Macintosh) MacOS 9.*x* or earlier limits we to 31 characters for filenames, but has no limit on characters for library names.

• We can transfer a VI library to other platforms more easily than we can transfer multiple individual VIs.

• We can slightly reduce the file size of our project because VI libraries are compressed to reduce disk space requirements.

• We can mark VIs in a library as top-level so when we open the library, LabVIEW automatically opens all the top-level VIs in that library.

7.6 SubVIs

After we build a VI and create its icon and connector pane, we can use it in another VI. A VI called from the block diagram of another VI is called a subVI. A subVI corresponds to a subroutine in text-based programming languages. A subVI

node corresponds to a subroutine call in text-based programming languages. The node is not the subVI itself, just as a subroutine call statement in a program is not the subroutine itself. A block diagram that contains several identical subVI nodes calls the same subVI several times.

The subVI controls and indicators receive data from and return data to the block diagram of the calling VI. Select VIs on the **Functions»Select a VI** palette and place them on the block diagram to create a subVI call to that VI.

7.6.1 Setting up the Connector Pane

To use a VI as a subVI, you need to build a connector pane, shown below. The connector pane is a set of terminals that corresponds to the controls and indicators of that VI, similar to the parameter list of a function call in text-based programming languages. The connector pane defines the inputs and outputs you can wire to the VI so you can use it as a subVI.

Fig 7.6.1.1Connector Pane

Define connections by assigning a front panel control or indicator to each of the connector pane terminals. To define a connector pane, right-click the icon in the upper right corner of the front panel window and select **Show Connector** from the shortcut menu. The connector pane replaces the icon. Each rectangle on the connector pane represents a terminal. Use the rectangles to assign inputs and outputs. The number of terminals LabVIEW displays on the connector pane depends on the number of controls and indicators on the front panel. The connector pane has, at most, 28 terminals. If your front panel contains more than 28 controls

and indicators that you want to use programmatically, group some of them into a cluster and assign the cluster to a terminal on the connector pane.

7.6.2 Creating an Icon

Every VI displays an icon, in the upper right corner of the front panel and block diagram windows. An icon is a graphical representation of a VI. It can contain text, images, or a combination of both. If we use a VI as a subVI, the icon identifies the subVI on the block diagram of the VI.

The default icon contains a number that indicates how many new VIs we have opened since launching LabVIEW. Create custom icons to replace the default icon by right-clicking the icon in the upper right corner of the front panel or block diagram and selecting **Edit Icon** from the shortcut menu or by double-clicking the icon in the upper right corner of the front panel. We also can drag a graphic from anywhere in our file system and drop it in the upper right corner of the front panel or block diagram. LabVIEW converts the graphic to a 32 × 32 pixel icon.

7.6.3 Creating SubVIs from Sections of a VI

Convert a section of a VI into a subVI by using the Positioning tool to select the section of the block diagram you want to reuse and selecting **Edit»Create SubVI**. An icon for the new subVI replaces the selected section of the block diagram. LabVIEW creates controls and indicators for the new subVI and wires the subVI to the existing wires.

Creating a subVI from a selection is convenient but still requires careful planning to create a logical hierarchy of VIs. Consider which objects to include in the selection and avoid changing the functionality of the resulting VI.

7.6.4 Loop and Case Structures

Structures are graphical representations of the loops and case statements of text-based programming languages. Use structures on the block diagram to repeat blocks of code and to execute code conditionally or in a specific order.

Like other nodes, structures have terminals that connect them to other block diagram nodes, execute automatically when input data are available, and supply data to output wires when execution completes.

Each structure has a distinctive, resizable border to enclose the section of the block diagram that executes according to the rules of the structure. The section of the block diagram inside the structure border is called a subdiagram. The terminals that feed data into and out of structures are called tunnels. A tunnel is a connection point on a structure border.

Use the following structures located on the **Functions»Structures** palette to control how a block diagram executes processes:

- **For Loop**—Executes a subdiagram a set number of times.
- **While Loop**—Executes a subdiagram until a condition is met.
- **Case structure**—Contains multiple subdiagrams, only one of which executes depending on the input value passed to the structure.
- **Sequence structure**—Contains one or more subdiagrams, which execute in sequential order.
- **Formula Node**—Performs mathematical operations based on numeric input.
- **Event Structure**—Contains one or more subdiagrams that execute depending on how the user interacts with the VI.

LabVIEW program is given in APPENDIX D

CHAPTER 8

WEB DESIGN

8.1 INTRODUCTION

Web Design is a process of conceptualization, planning, modeling, and execution of electronic media content delivery via Internet in the form of technologies (such as markup languages) suitable for interpretation and display by a web browser or other web-based graphical user interfaces (GUIs).

8.2 ELEMENTS OF A WEB PAGE

A web page, as an information set, can contain many kinds of information, which is able to be seen, heard or interact by the end user:

Perceived (rendered) information:

- **Textual information**: with diverse render variations.
- **Non-textual information**:
    - Static images on raster graphics, typically GIF, JPEG or PNG; or vector formats as SVG or Flash.
    - Animated images typically Animated GIF and SVG, but also may be Flash, Shockwave, or Java applet.
    - Audio, typically MIDI or WAV formats or Java applets.
    - Video, WMV (Windows), RM (Real Media), FLV (Flash Video), MPG, MOV (Quicktime)
- **Interactive information**: more complex, glued to interface.
    - For "on page" interaction:
        - **Interactive text**

        - **Interactive illustrations**: ranging from "click to play" image to games, typically using script orchestration, Flash, Java applets, SVG, or Shockwave.
        - **Buttons**: forms providing alternative interface, typically for use with script orchestration and DHTML.
    - For "between pages" interaction:
        - **Hyperlinks**: standard "change page" reactivity.
        - **Forms**: providing more interaction with the server and server-side databases.

Internal (hidden) information:

- Comments
- Metadata with semantic meta-information, Charset information, Document Type Definition (DTD), etc.
- Diagramation and style information: information about rendered items (like image size attributes) and visual specifications, as Cascading Style Sheets (CSS).
- Scripts, usually JavaScript, complement interactivity and functionality.

The web page can also contain dynamically adapted information elements, dependent upon the rendering browser or end-user location (through the use of IP address tracking and/or "cookie" information).

From a more general/wide point of view, some information (grouped) elements, like a navigation bar, are uniform for all website pages, like a standard. These kind of "website standard information" are supplied by technologies like web template systems.

8.3 URL

Typically, web pages today are becoming more dynamic. A dynamic web page is one that is created server-side when it is requested, and then served to the end-user. These types of web pages typically do not have a permalink, or a static URL, associated with them. Today, this can be seen in many popular forums, online shopping. This practice is intended to reduce the amount of static pages in lieu of storing the relevant web page information in a database. Some search engines may have a hard time indexing a web page that is dynamic, so static web pages can be provided in those instances.

8.4 VIEWING A WEB PAGE

In order to graphically display a web page, a web browser is needed. This is a type of software that can retrieve web pages from the Internet. Most current web browsers include the ability to view the source code. Viewing a web page in a text editor will also display the source code, not the visual product.

8.5 CREATING A WEB PAGE:

To create a web page, a text editor or a specialized HTML editor is needed. In order to upload the created web page to a web server, traditionally an FTP client is needed.

The design of a web page is highly personal. A design can be made according to one's own preference, or a pre-made web template can be used. Web Templates let web page designers edit the content of a web page without having to worry about the overall aesthetics. Many people publish their own web pages using products like Geocities from Yahoo, Tripod, or Angelfire. These web publishing tools offer free page creation and hosting up to a certain size limit.

Other ways of making a web page is to download specialized software, like a Wiki, CMS, or forum. These options allow for quick and easy creation of a web page which is typically dynamic. Wikipedia, WordPress, and Invision Power Board are examples of the above three web page options.

8.6 SAVING A WEB PAGE

While one is viewing a web page, a copy of it is saved locally; this is what is being viewed. Depending on the browser settings, this copy may be deleted at any time, or stored indefinitely, sometimes without the user realizing it. Most GUI browsers will contain all the options for saving a web page more permanently. These include, but are not limited to:

- Saving the rendered text without formatting or images - Hyperlinks are not identified, but displayed as plain text
- Saving the HTML file as it was served - Overall structure will be preserved, although some links may be broken
- Saving the HTML file and changing relative links to absolute ones - Hyperlinks will be preserved
- Saving the entire web page - All images will be saved, as well as links being changed to absolute
- Saving the HTML file including all images, stylesheets and scripts into a single MHTML file. This is supported by Internet Explorer, Mozilla, Mozilla Firefox and Opera. Mozilla and Mozilla Firefox only support this if the MAF plugin has been installed. An MHTML file is based upon the MHTML standard.

Common web browsers, like Mozilla Firefox, Internet Explorer and Opera, give the option to not only print the currently viewed web page to a printer, but

optionally to "print" to a file which can be viewed or printed later. Some web pages are designed, for example by use of CSS, so that hyperlinks, menus and other navigation items, which will be useless on paper, are rendered into print with this in mind. Space-wasting menus and navigational blocks may be absent from the printed version; other hyperlinks may be shown with the link destinations made explicit, either within the body of the page or perhaps listed at then end.

8.7 ASP.NET

ASP.NET is a web application framework marketed by Microsoft that programmers can use to build dynamic web sites, web applications and web services. It is part of Microsoft's .NET platform and is the successor to Microsoft's Active Server Pages (ASP) technology. ASP.NET is built on the Common Language Runtime, allowing programmers to write ASP.NET code using any Microsoft .NET language.

8.8 ASPX FILE format

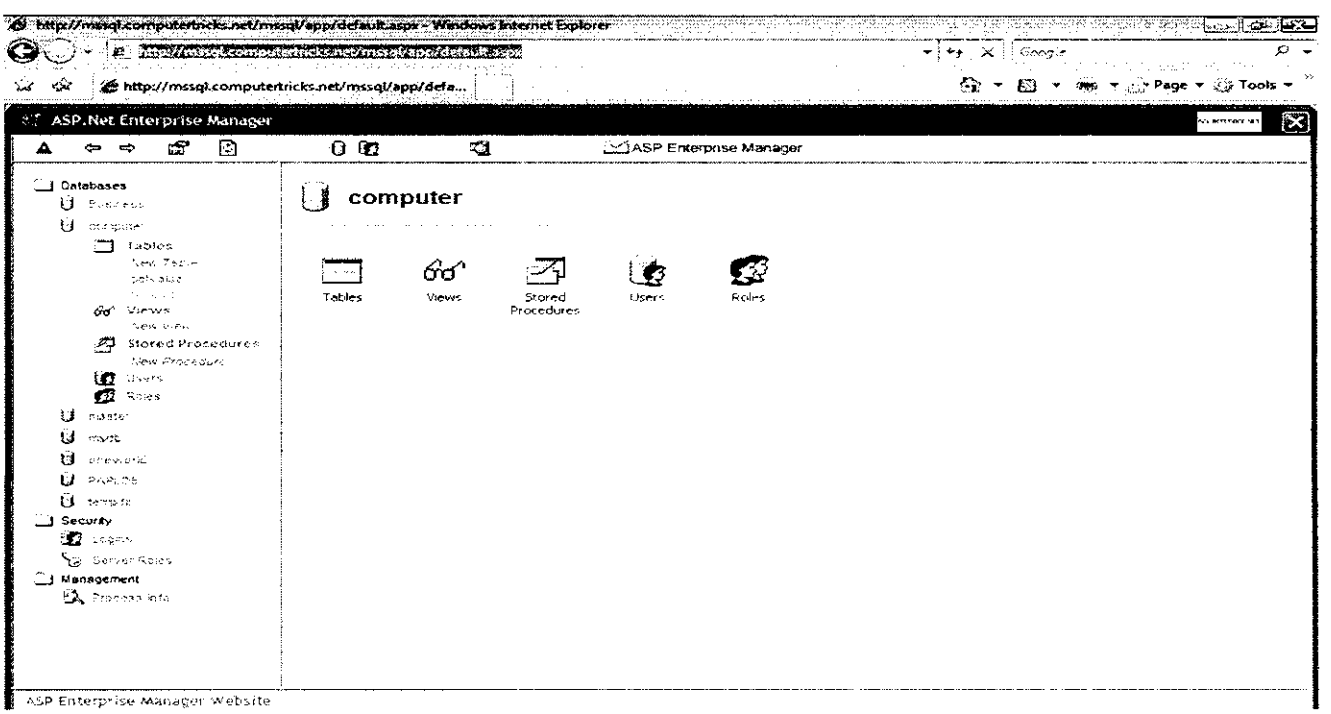
ASPX is an html file format used to create Webform pages; in programming jargon, the ASPX file typically contains static HTML or XHTML markup, as well as markup defining Web Controls and Web User Controls where the developers place all the required static and dynamic content for the web page. Additionally, dynamic code which runs on the server can be placed in a page within a block **<% -- dynamic code -- %>** which is similar to other web development technologies such as PHP, JSP, and ASP, but this practice is generally frowned upon by Microsoft except for the purposes of data binding since it requires more calls when rendering the page.

It is recommended by Microsoft for dealing with dynamic program code to use the code-behind model, which places this code in a separate file or in a

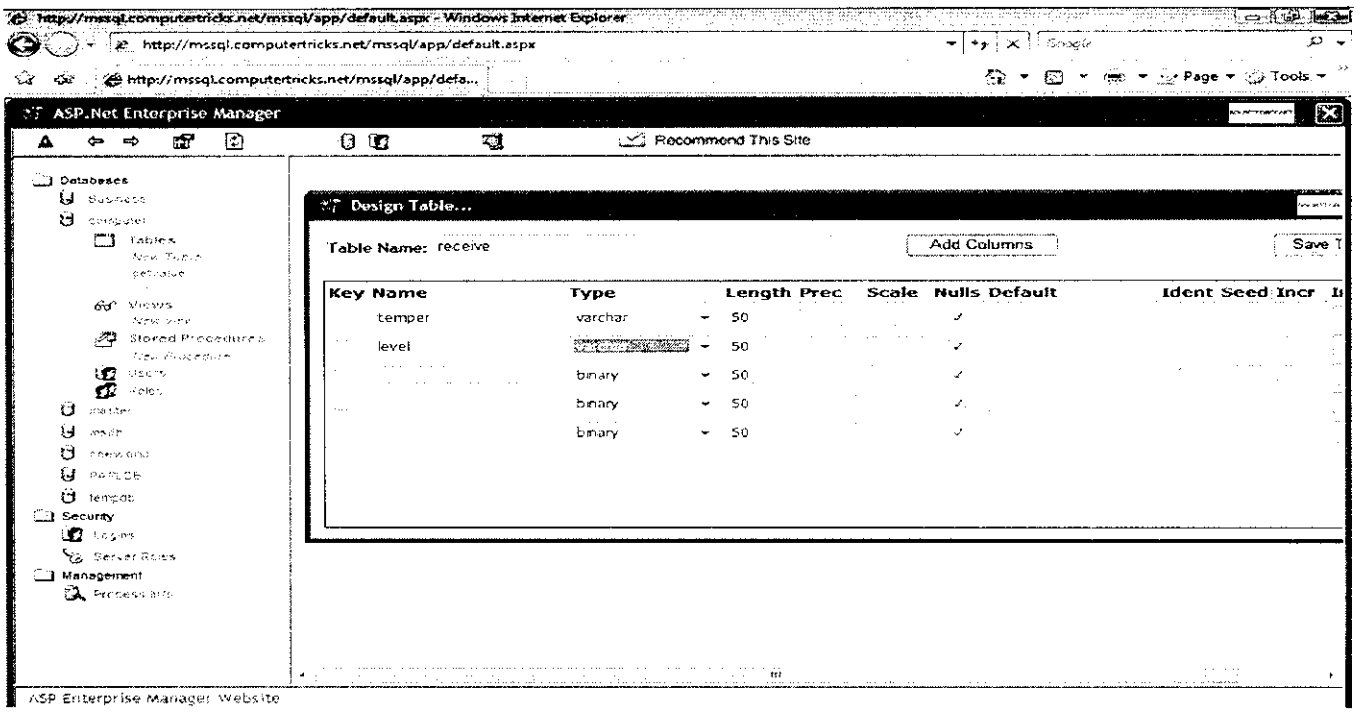
specially designated script tag. Code-behind files are typically named something to the effect of *MyPage.aspx.cs* or *MyPage.aspx.vb* based on the ASPX file name (this practice is automatic in Microsoft Visual Studio and other IDEs). When using this style of programming, the developer writes code to respond to different events, like the page being loaded, or a control being clicked, rather than a procedural walk through the document.

ASP.NET's code-behind model marks a departure from other web programming languages (including Classic ASP) in that it attempts to separate the design code from the programming code itself. In theory, this would allow a web designer, for example, to focus on the design markup with less potential for disturbing the programming code that drives it.

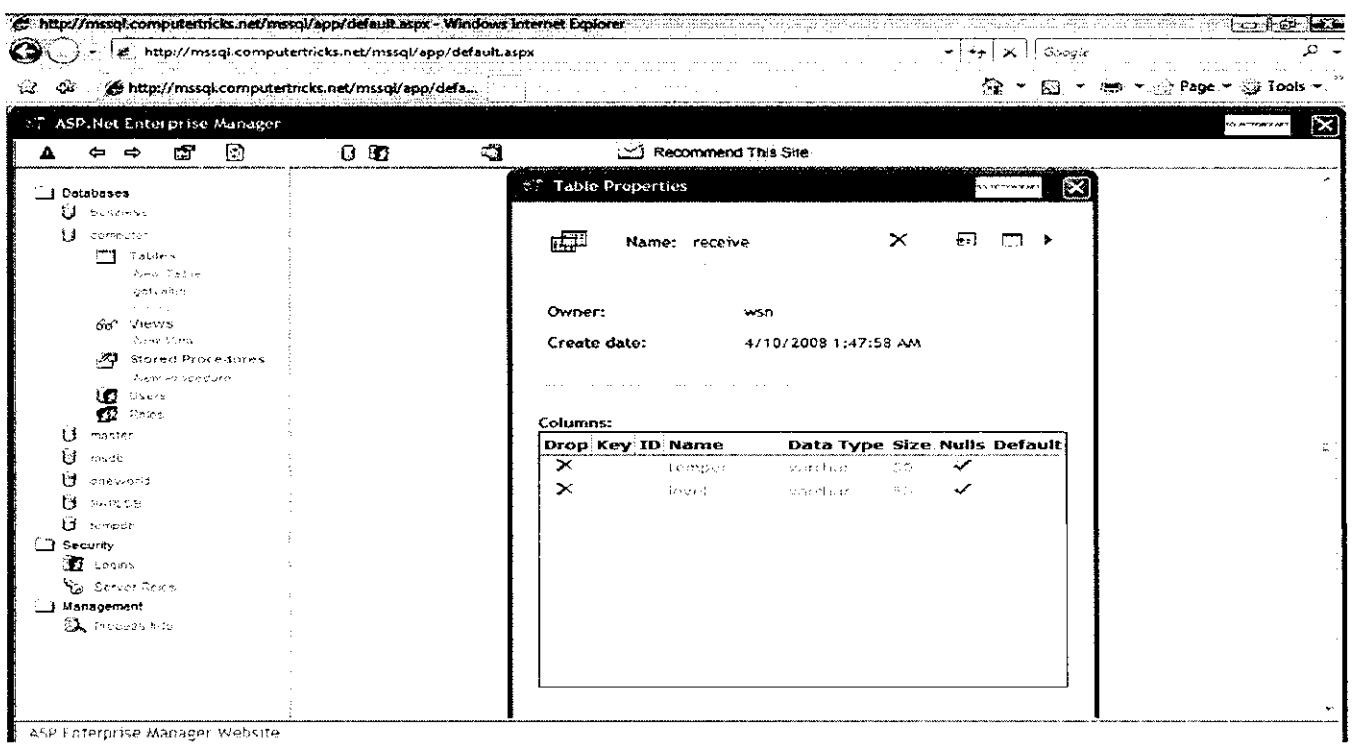
Steps in Creating SQL Database Table



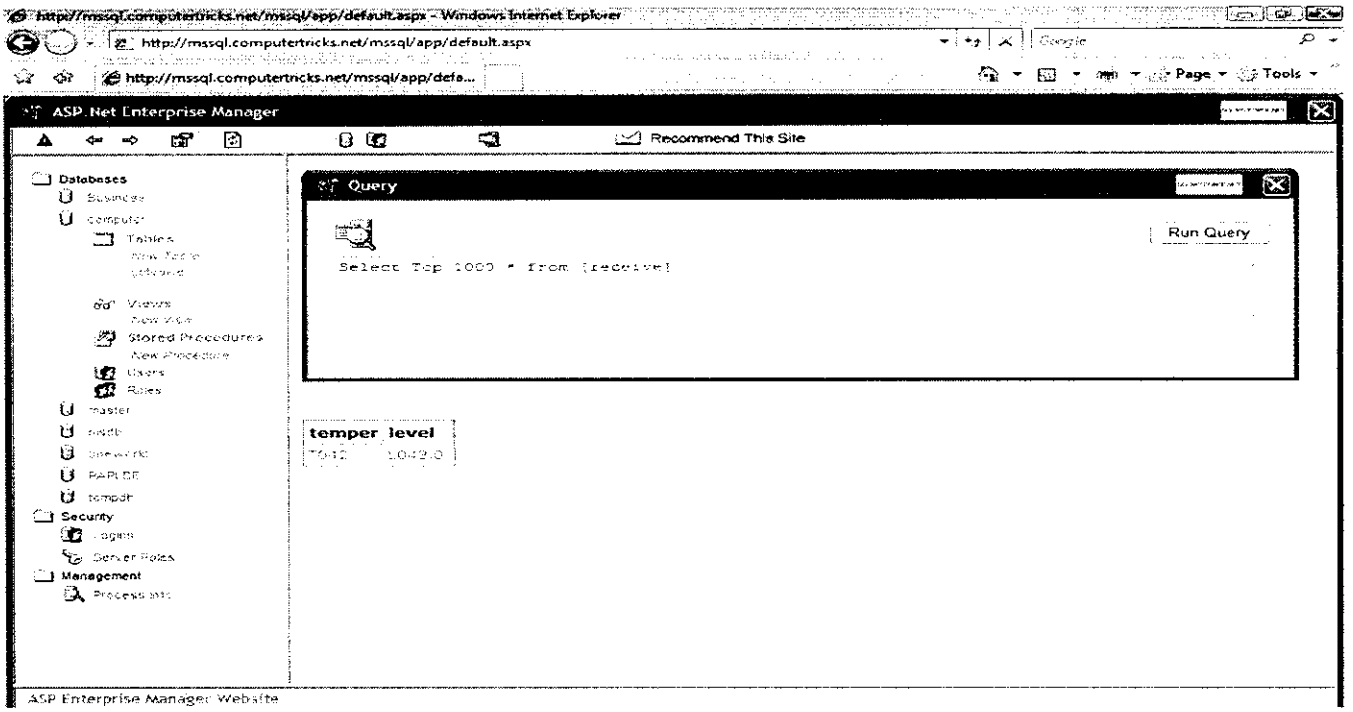
Step 1: Home page of database 'Computer'



Step 2: Creating a Table from table option



Step 3: Table properties for created table 'receive'



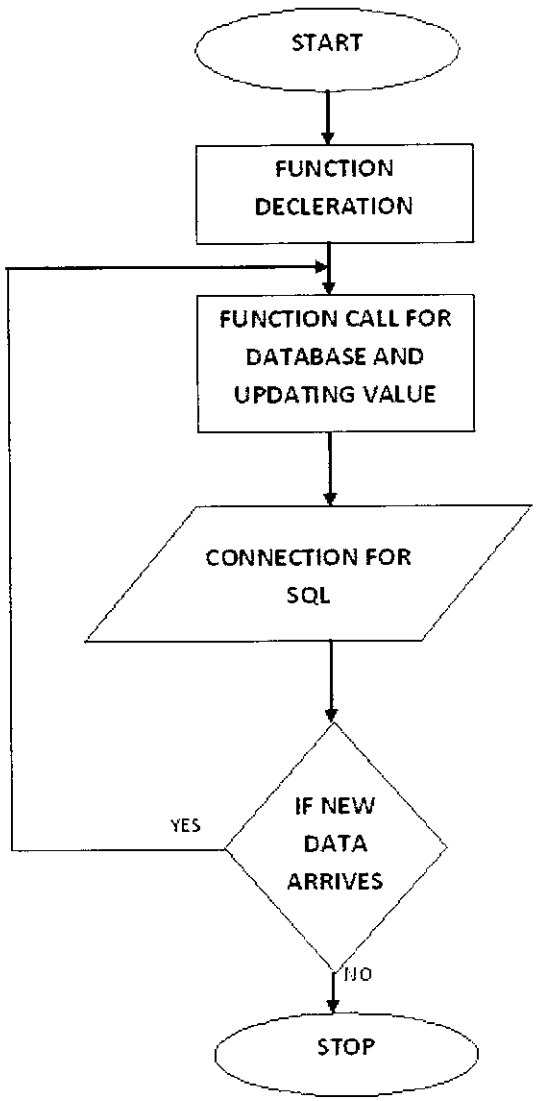
Step 4: Initialize the values for the table

8.9 SERVICE7

8.9.1 ALGORITHM

- All the procedures required for accessing the web service are initialized
- Temperature and Level values are given to function calling the database
- SQL connection is made for the specified database
- The database table is updated every time when the function is called
- The table is opened and the SQL command is executed to append the values to the table and then it is closed.

8.9.2 FLOWCHART

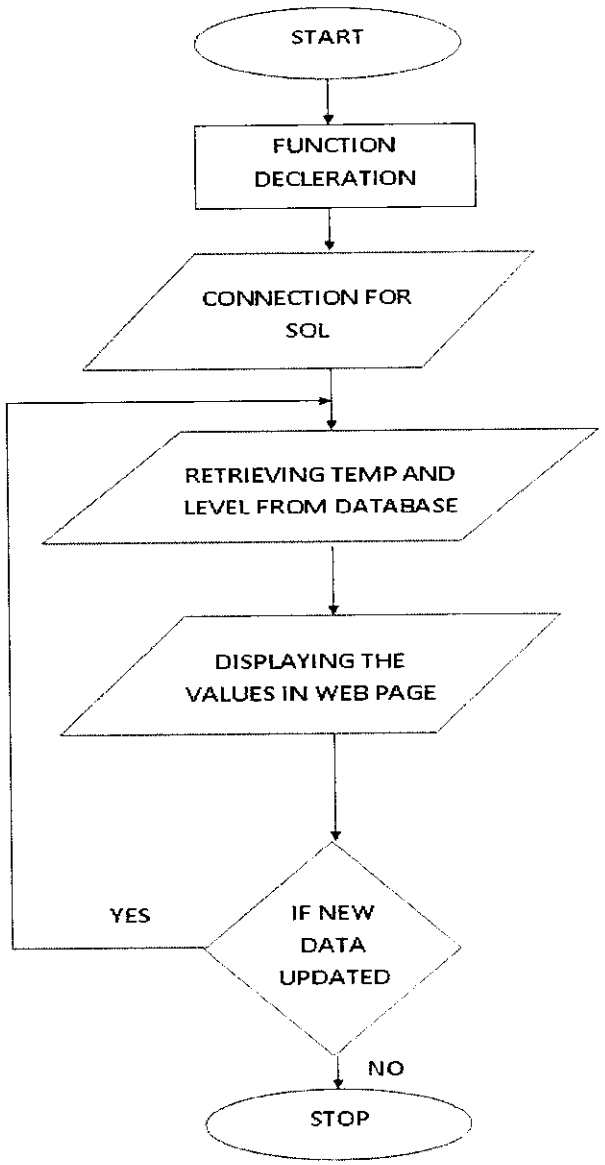


8.10 RECEIVE

8.10.1 ALGORITHM

- Initialize all the required variables
- Establishing the SQL connection with server name, userid, password and database name
- Retrieving the temperature and level value from the database table
- Display the values in the Text box
- Periodically the page is refreshed for displaying the new values

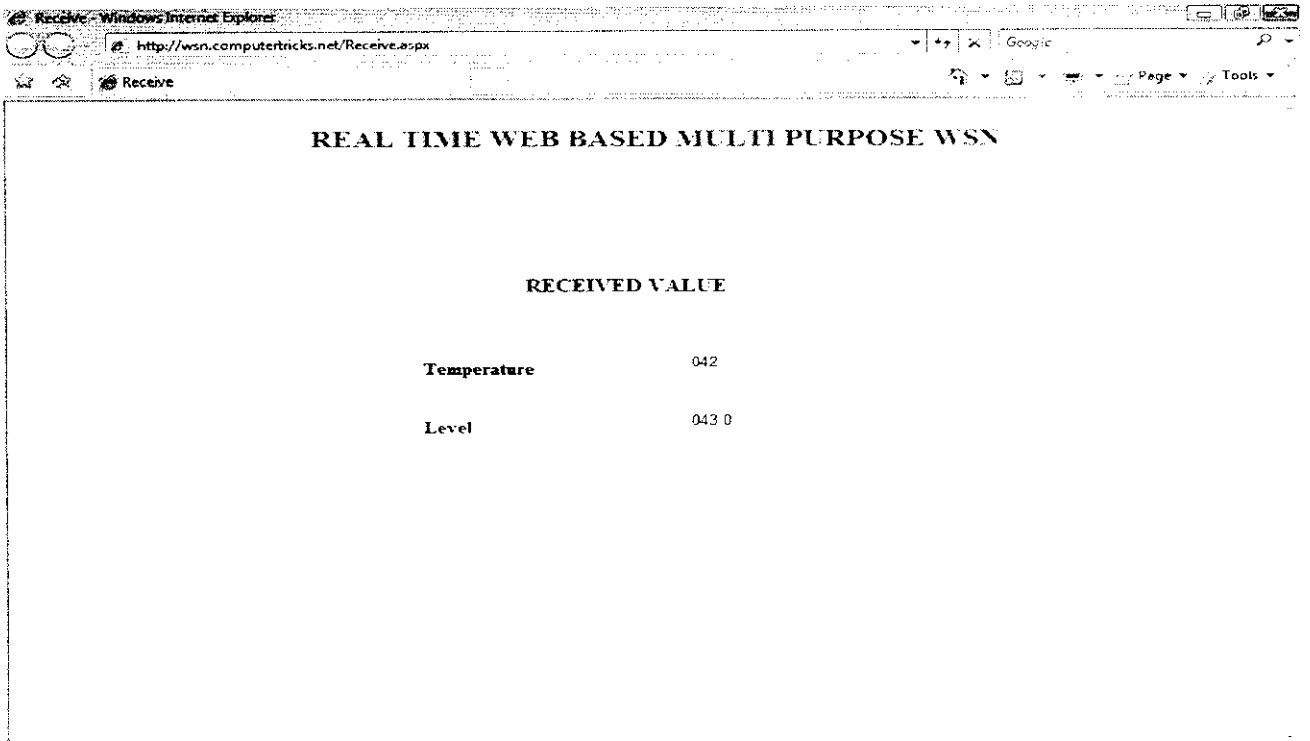
8.10.2 FLOWCHART



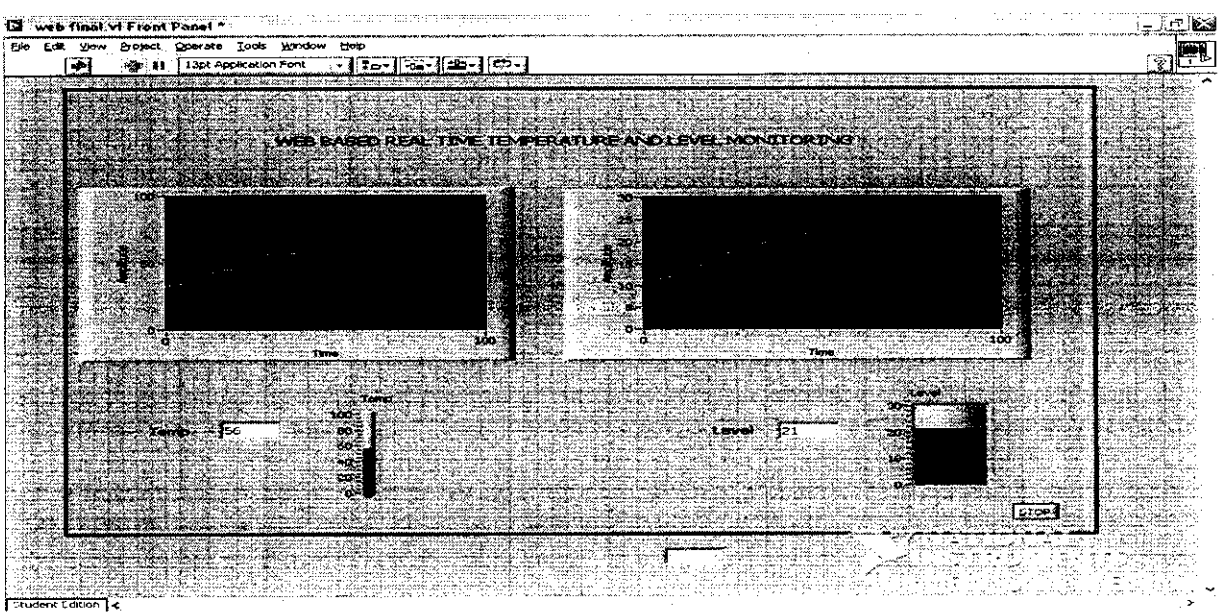
Further details about programming is given in APPENDIX E.

CHAPTER 9

RESULTS



Web Page displaying output values



Output screen on LabVIEW

CHAPTER 10
CONCLUSION

In our system we are using three parts real time sensing, wireless transmission, uploading into website. Since we are using wireless the server need not to be nearer to the boiler. The parameters are sensed and transmitted via wireless to the server. The manual work is also reduced. The temperature and level measurements are monitored from anywhere in this world. In future this system can be expanded by adding controller part after monitoring. Also it can be expanded by adding more sensors for monitoring more parameters.

APPENDIX A

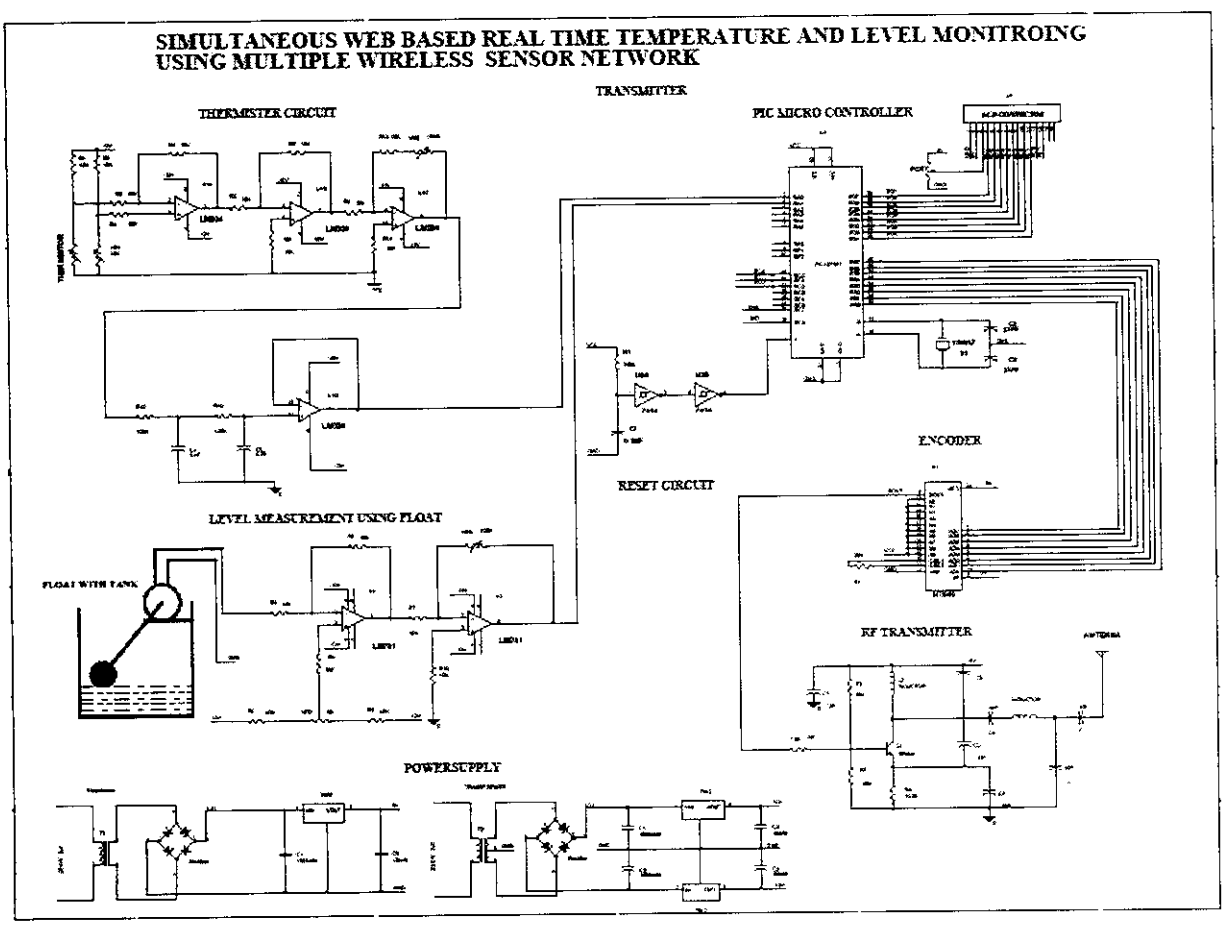


Fig Schematic diagram of Transmitter part

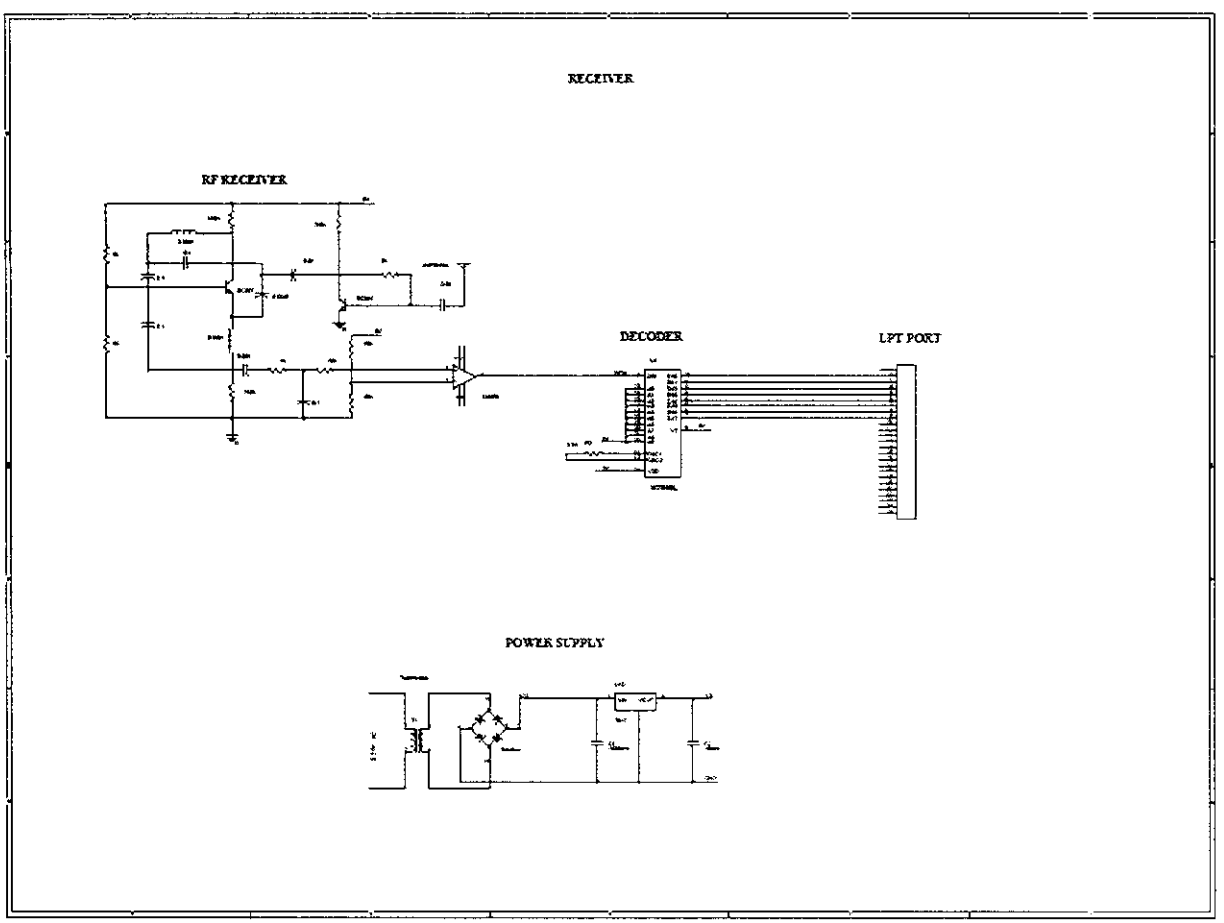


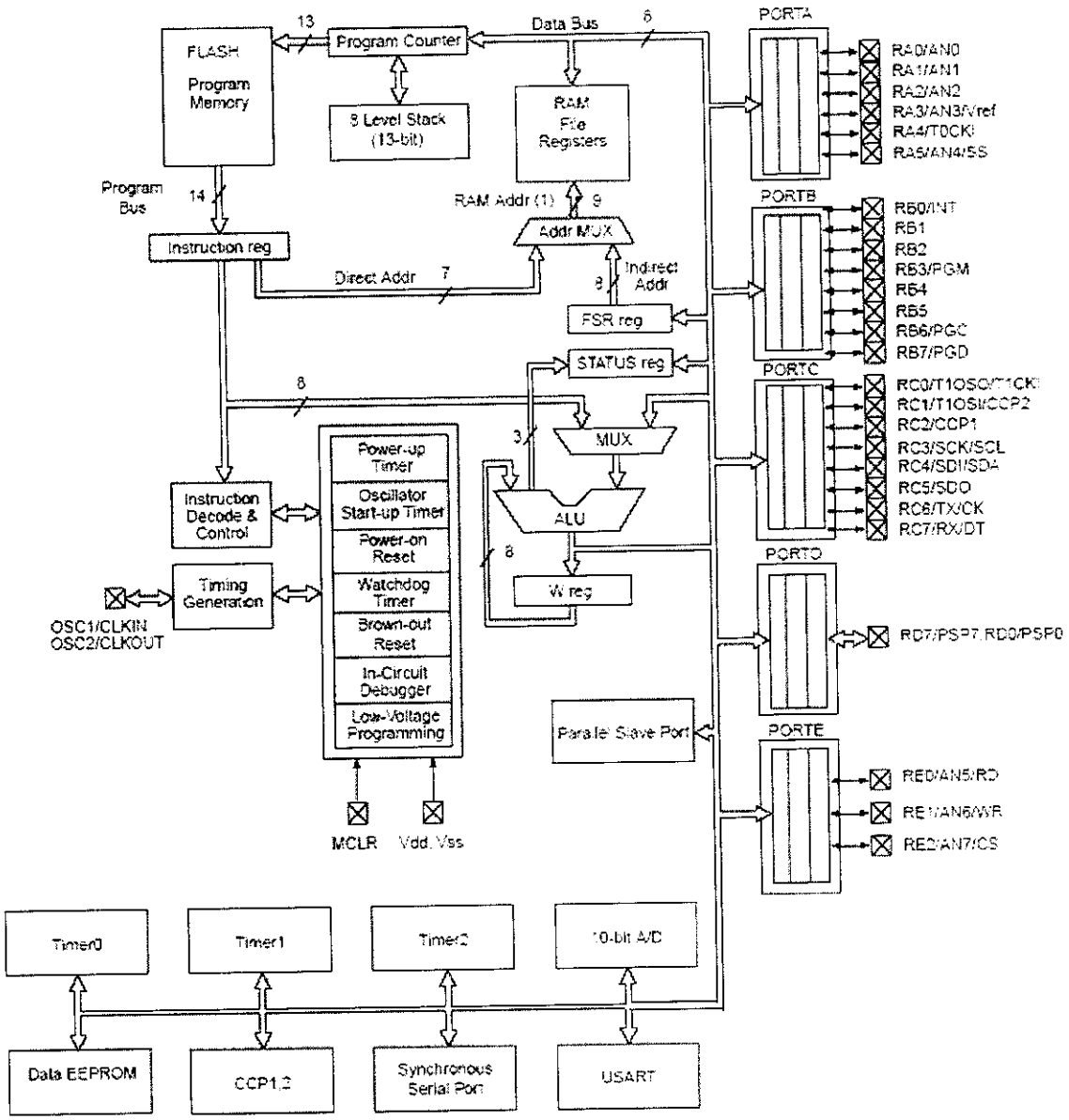
Fig Schematic diagram of Receiver part

APPENDIX B

PERIPHERAL FEATURES

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during sleep
  Via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  Capture is 16-bit, max resolution is 12.5 ns,
  Compare is 16-bit, max resolution is 200 ns,
  PWM max. Resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI. (Master Mode) and I2C. (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9- Bit addresses detection.
- Brown-out detection circuitry **for Brown-out Reset (BOR)**

ARCHITECTURE OF PIC 16F877

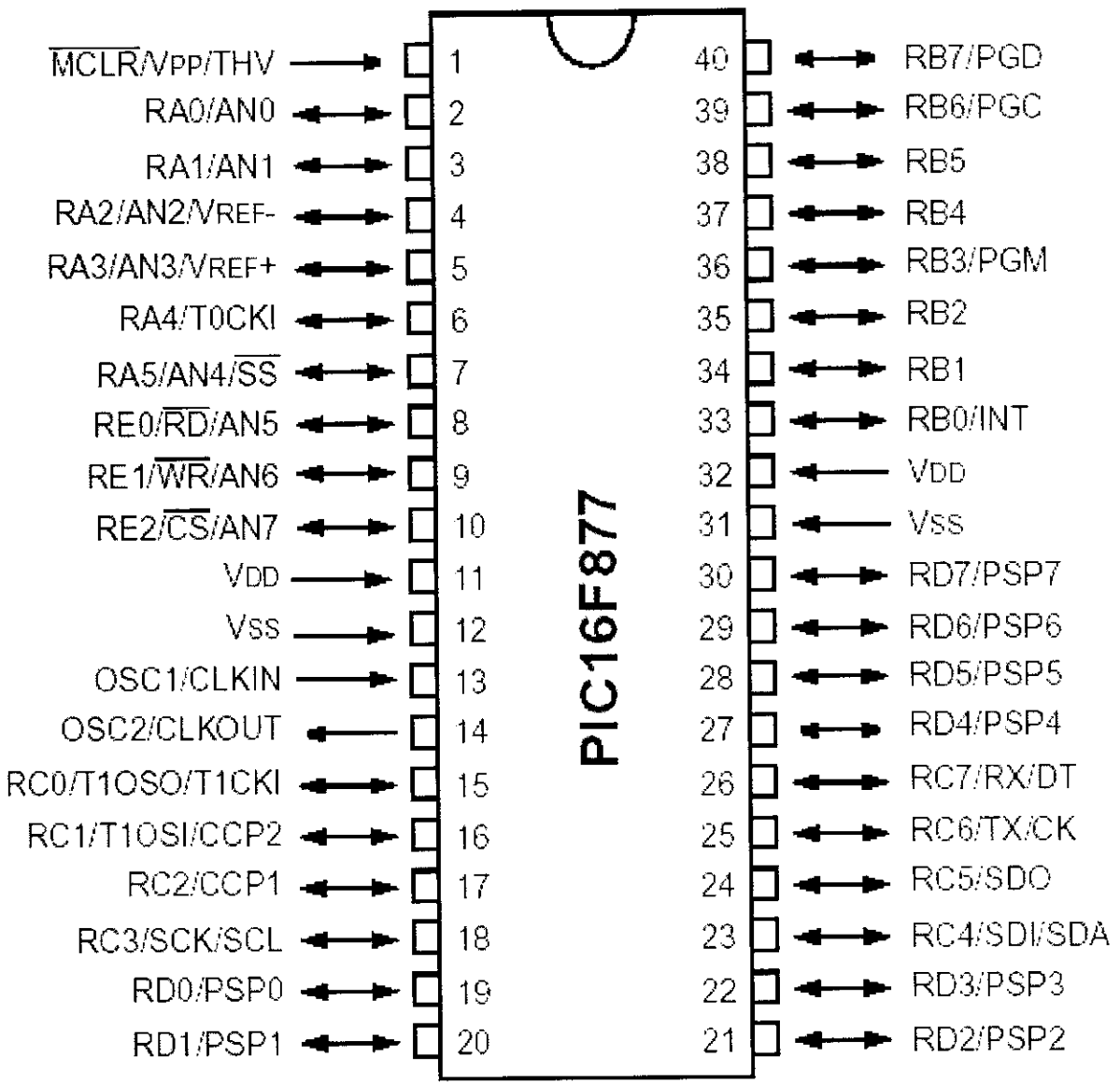


Note 1: Higher order bits are from the STATUS register.

TABLE SPECIFICATIONS

DEVICE	PROGRAM FLASH	DATA MEMORY	DATA EEPROM
PIC 16F877	8K	368 Bytes	256 Bytes

PIN DIAGRAM OF PIC 16F877



PIN OUT DESCRIPTION

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS(4)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
MCLR/VPP/THV	1	2	18	I/P	ST	Master clear (reset) input or programming voltage input or high voltage test mode control. This pin is an active low reset to the device.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19	I/O	TTL	RA0 can also be analog input0
RA1/AN1	3	4	20	I/O	TTL	RA1 can also be analog input1
RA2/AN2/VREF-	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage
RA3/AN3/VREF+	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage
RA4/T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/counter. Output is open drain type.
RA5/SS/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input
RB4	37	41	14	I/O	TTL	Interrupt on change pin.
RB5	38	42	15	I/O	TTL	Interrupt on change pin.
RB6/PGC	39	43	16	I/O	TTL/ST(2)	Interrupt on change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST(2)	Interrupt on change pin or In-Circuit Debugger pin. Serial programming data.

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
						PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	RC0 can also be the Timer1 oscillator output or a Timer1 clock input.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	17	19	36	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	18	20	37	I/O	ST	RC3 can also be the synchronous serial clock input/ output for both SPI and I^2C modes.
RC4/SDI/SDA	23	25	42	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	24	26	43	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	25	27	44	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	26	29	1	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
						PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	I/O	ST/TTL(3)	
RD1/PSP1	20	22	39	I/O	ST/TTL(3)	
RD2/PSP2	21	23	40	I/O	ST/TTL(3)	
RD3/PSP3	22	24	41	I/O	ST/TTL(3)	
RD4/PSP4	27	30	2	I/O	ST/TTL(3)	
RD5/PSP5	28	31	3	I/O	ST/TTL(3)	
RD6/PSP6	29	32	4	I/O	ST/TTL(3)	
RD7/PSP7	30	33	5	I/O	ST/TTL(3)	
						PORTE is a bi-directional I/O port.
RE0/$\overline{RD}$/AN5	8	9	25	I/O	ST/TTL(3)	RE0 can also be read control for the parallel slave port, or analog input5.
RE1/$\overline{WR}$/AN6	9	10	26	I/O	ST/TTL(3)	RE1 can also be write control for the parallel slave port, or analog input6.
RE2/$\overline{CS}$/AN7	10	11	27	I/O	ST/TTL(3)	RE2 can also be select control for the parallel slave port, or analog input7.
V_{SS}	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
V_{DD}	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17,28, 40	12,13, 33,34		—	These pins are not internally connected. These pins should be left unconnected.

Legend: I = input O = output I/O = input/output P = power

-= Not used TTL = TTL input ST = Schmitt Trigger input

OSCILLATOR AND CLOCK CIRCUIT

XTAL1 and XTAL2 are the input and output respectively of an inverting amplifier which is intended for use as a crystal oscillator in the pioerce configuration, in the frequency range of 1.2 MHz to 12 MHz. XTAL2 also the input to the internal clock generator. To drive the chip with an internal oscillator, one would ground XTAL1 and XTAL2. Since the input to the clock generator is divide by two flip flops there are no requirements on the duty cycle of the external oscillator signal. However, minimum high and low times must be observed.

The clock generator divides the oscillator frequency by 2 and provides a tow phase clock signal to the chip. The phase 1 signal is active during the first half to each clock period and the phase 2 signals are active during the second half of each clock period.

ADCON0 REGISTER (ADDRESS: 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON

R = Readable bit

W = Writable bit

U = Unimplemented bit,read as '0'

- n = Value at POR reset

bit 7-6: **ADCS1:ADCS0: A/D Conversion Clock Select bits**

00 = FOSC/2

01 = FOSC/8

10 = FOSC/32

11 = FRC (clock derived from an RC oscillation)

bit 5-3: **CHS2:CHS0**: Analog Channel Select bits

000 = channel 0, (RA0/AN0)

001 = channel 1, (RA1/AN1)

010 = channel 2, (RA2/AN2)

011 = channel 3, (RA3/AN3)

100 = channel 4, (RA5/AN4)

101 = channel 5, (RE0/AN5)**(1)**

110 = channel 6, (RE1/AN6)**(1)**

111 = channel 7, (RE2/AN7)**(1)**

bit 2: **GO/DONE:** A/D Conversion Status bit

If ADON = 1

1 = A/D conversion in progress (setting this bit starts the A/D conversion)

0 = A/D conversion not in progress (This bit is automatically cleared by hardware when the A/D conversion is complete)

bit 1: **Unimplemented**: Read as '0'

bit 0: **ADON**: A/D On bit

1 = A/D converter module is operating

0 = A/D converter module is shutoff and consumes no operating current

ADCON1 REGISTER (ADDRESS 9Fh)

U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0

bit 7: **ADFM: A/D Result format select**

1 = Right Justified. 6 most significant bits of ADRESH are read as '0'.

0 = Left Justified. 6 least significant bits of ADRESL are read as '0'.

bit 6-4: **Unimplemented: Read as '0'**

bit 3-0: **PCFG3:PCFG0**: A/D Port Configuration Control bits

MICROCONTROLLER PROGRAMING

```
#include<pic.h>
#include<lcd.h>
void adc_init(void);
void adc0(void);
void adc1(void);
void hex_dec_fuel(unsigned char);

unsigned char bri,sol1,sol2,j;
unsigned int temp0,temp1;
void main()
{
TRISC = 0x0C;
TRISB = 0x00;

adc_init();
lcd_init();

command(0x80);
lcd_condis(" Web Based Temp ",16);
command(0xc0);
lcd_condis("& Level Monitor.",16);
delay(60000);delay(60000);
```

```
command(0x80);
lcd_condis(" Web Based Mon. ",16);
command(0xc0);
lcd_condis("T:000  L:00.0   ",16);
while(1)
{
adc0();
command(0xc2);
hex_dec(sol1);
adc1();
command(0xc9);
hex_dec_fuel(sol2);

PORTB=0xff;
delay(40000);
PORTB='*';
delay(40000);
PORTB=0xff;
delay(40000);
PORTB=sol1;
delay(40000);
PORTB=0xff;
delay(40000);
PORTB=sol2;
delay(40000);
PORTB=0xff;
delay(40000);
```

```
}                                   //while

}                                   //main

void adc_init()
{
 ADCON1=0x02;                       // 5-channel, Left justified, ADC control
 TRISA=0xff;                        // to select the port A as input port
}

void adc0()
{
temp0=0;
for(j=0;j<5;j++)
 {
   ADCON0=0x00;                     // Channel select (Cha: 0)
   ADON=1;                          // ADC module ON
   delay(255);
   ADCON0 =0x05;                    // selecting a channel 0 and making the go/done bit
                                    high
   while(ADCON0!=0X01);             // Chk whether conversion finished or not
   temp1 = ADRESH;                  // 8 bit value taken into one variable
   temp0 = temp0 + temp1;
 }
   sol1=temp0/5;
}

void adc1()X{
```

```
temp0=0;
for(j=0;j<5;j++)
 {
   ADCON0=0x08;
   ADON=1;
   delay(255);
   ADCON0 =0x0d;            // selecting a channel 1 and making the go/done bit
                               high
   while(ADCON0!=0x09);
   temp1 = ADRESH;
   temp0 = temp0 + temp1;
 }
sol2=temp0/5;
}

void hex_dec_fuel(unsigned char vai)
{
 h=vai/100;
 hr=vai%100;
 t=hr/10;
 o=hr%10;

 lcd_disp(h+0x30);
 lcd_disp(t+0x30);
 lcd_disp('.');
 lcd_disp(o+0x30);
 }
```

APPENDIX C

ELECTRICAL CHARACTERISTICS OF RF TRANSMITTER

Parameter	Sym.	Min.	Typ.	Max.	Unit
Operating Frequency (200KHz)	Vcc		433.92		MHz
Data Rate	ASK			8K	Kbps
Transmitter Performance(OOK@2.4kbps)					
Peak Input Current.12 Vdc Supply	ITP			45	mA
Peak Output Power	PO		10		mW
Tum On/ Tum Off Time	T ON/T OFF			1	US
Power Supply Voltage Range	Vcc	3		12	VDC
Operating Ambient Temperature	TA	-20		+85	centigrade
Tx Antenna Out (3V) +2.4dB	Vcc				mA

ABSOLUTE MAXIMUM RATINGS

Rating	Value	Unit
Power Supply and All Input/Output Pins	-0.3 to +12.0	V
Non-Operating Case Temperature	-20 to +85	centigrade
Soldering Temperature (10 seconds)	230	centigrade

DC CHARACTERISTICS OF RF RECEIVER

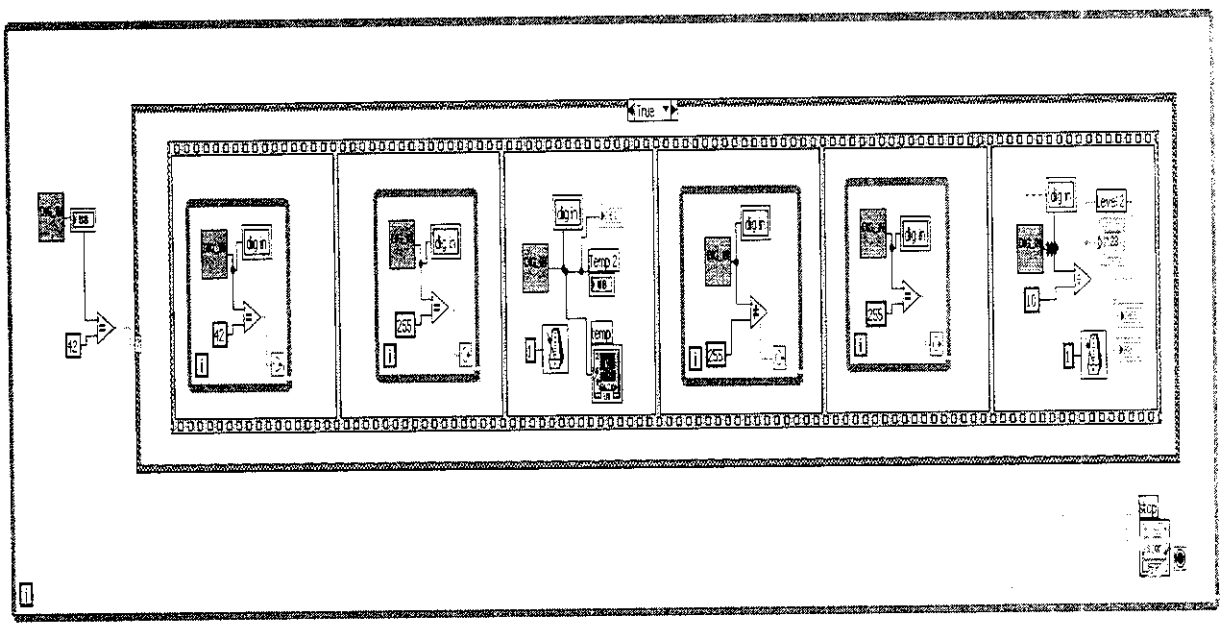
Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
Vcc	Operating Supply Voltage		4.9	5	5.1	
I Tot	Operating Supply Voltage			4.5		
V Data	Data Out	1 Data = +200 uA (High)	Vcc -0.5	Vcc		V
		1 Data = -10 uA (Low)			0.3	V

ELECTRICAL CHARACTERISTICS OF RF RECEIVER

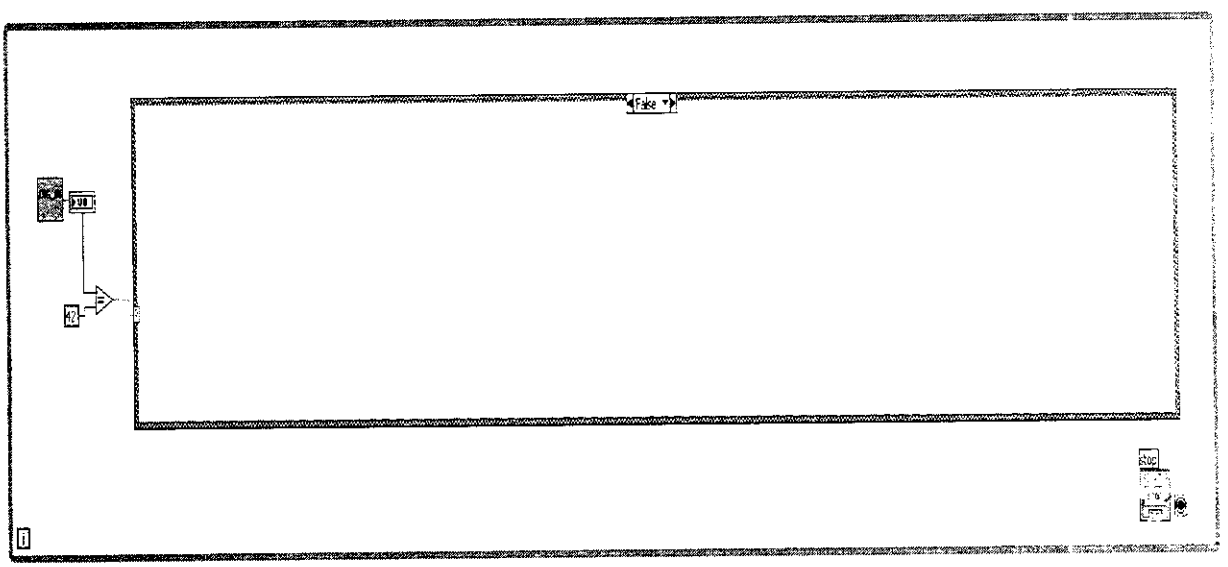
Parameter	Sym.	Min.	Typ.	Max.	Unit
Operating Radio Frequency	FC		433.92		MHz
Sensitivity	Pref.		-108		dBm
Channel Width		-500		+500	KHz
Noise Equivalent BW	NEB		5	4	KHz
Baseboard Data Rate			3	KB/S	
Receiver Tum On Time				3	ms

APPENDIX D

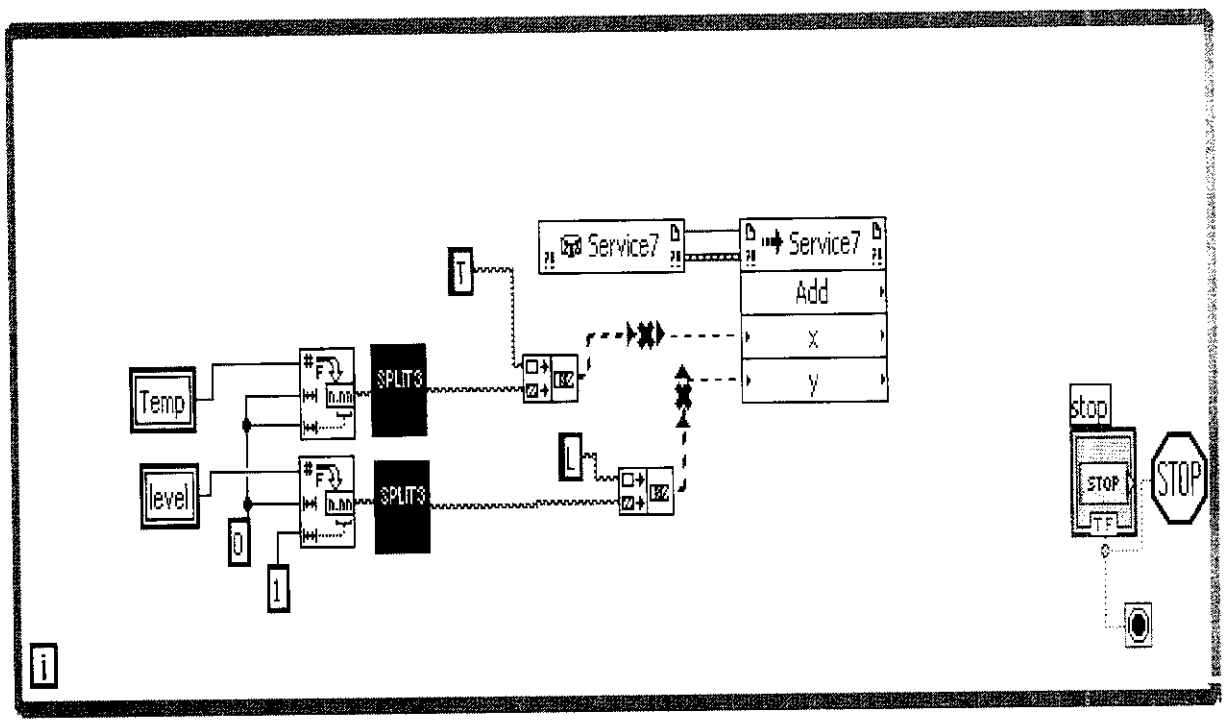
LabVIEW BLOCK DIAGRAM



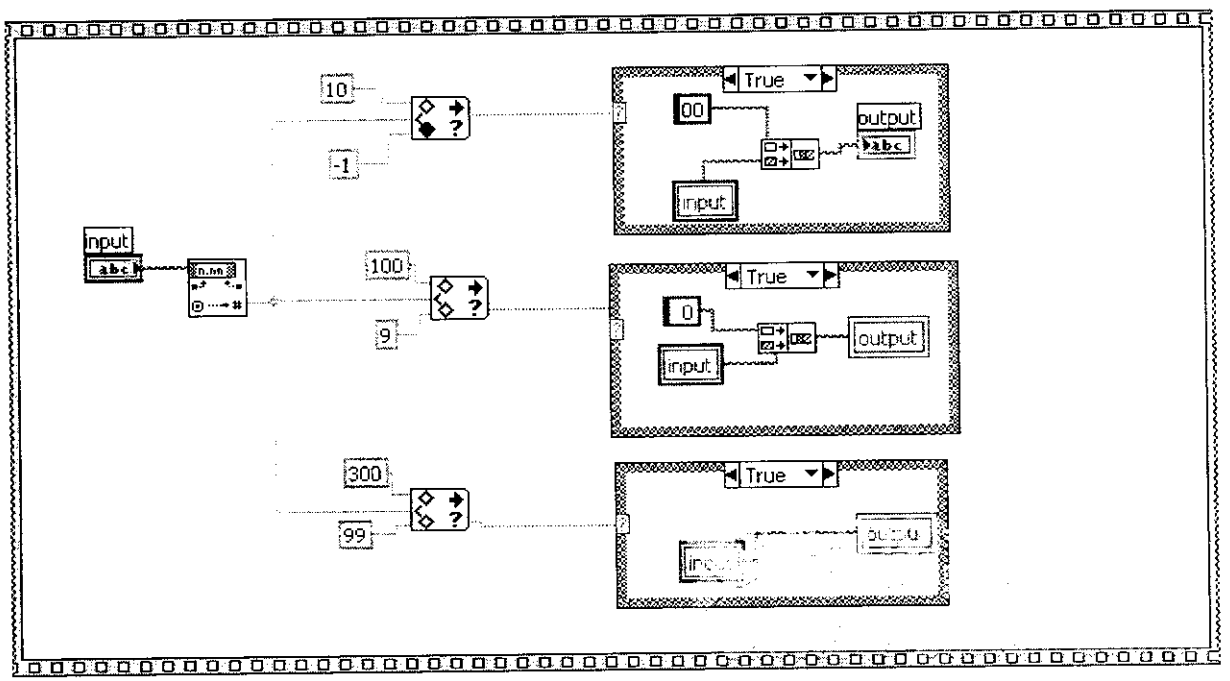
If cond true



If cond false



Web accessing



Split3

APPENDIX E

SERVICE7

```
<%@ WebService Language="vb" Class="Service7" %>
Imports System.Data
Imports System.Data.SqlClient
Imports System.Web.Services

<System.Web.Services.WebService(Namespace:="http://tempuri.org/labview/Serv
ice7")> _
Public Class Service7
   Inherits System.Web.Services.WebService
#Region " Web Services Designer Generated Code "
   Public Sub New()
   MyBase.New()
      'This call is required by the Web Services Designer.
   InitializeComponent()
         'Add your own initialization code after the InitializeComponent() call
   End Sub
         'Required by the Web Services Designer
   Private components As System.ComponentModel.IContainer
         'NOTE: The following procedure is required by the Web Services Designer
         'It can be modified using the Web Services Designer.
         'Do not modify it using the code editor.
   <System.Diagnostics.DebuggerStepThrough()> Private Sub
   InitializeComponent()
      components = New System.ComponentModel.Container
```

```
End Sub

Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        'CODEGEN: This procedure is required by the Web Services Designer
        'Do not modify it using the code editor.
If disposing Then
    If Not (components Is Nothing) Then
        components.Dispose()
      End If
End If
      MyBase.Dispose(disposing)
End Sub

#End Region
  <WebMethod()> Public Function Add(ByVal x As String, ByVal y As String)
As String

      Dim sql As String
      Dim con As SqlConnection = New
SqlConnection("server=wsn.computertricks.net;uid=wsn;password=cheran;databas
e=computer")
      Dim com As SqlCommand
      sql = "UPDATE receive SET temper='" + x + "',level='" + y + "'"
      com = New SqlCommand(sql, con)
      con.Open()
      com.ExecuteNonQuery()
      con.Close()
      Return ("Successfully Updated")
```

```
    End Function
End Class
```

RECEIVE 7

```
<%@ import Namespace="System.Data.SqlClient"%>
<%@ Import Namespace="System.Data"%>
<%@ Import Namespace="System"%>
<%@ Page Language="vb" Debug=true%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
        <HEAD>
                <title>Receive</title>
                <meta content="Microsoft Visual Studio .NET 7.1"
                  name="GENERATOR">
                <meta content="Visual Basic .NET 7.1"
                  name="CODE_LANGUAGE">
                <meta content="JavaScript" name="vs_defaultClientScript">
                <meta content="http://schemas.microsoft.com/intellisense/ie5"
name="vs_targetSchema">
                <meta http-equiv="refresh" content="1;URL=Receive.aspx">
        </HEAD>
        <body MS_POSITIONING="GridLayout">
                <form id="Form1" method="post" runat="server">
                        <asp:textbox id="TextBox1" style="Z-INDEX: 103; LEFT:
536px; POSITION: absolute; TOP: 240px"
```

```
                runat="server" Width="144px"
Height="36px"></asp:textbox><asp:label id="Label4" style="Z-INDEX: 107;
LEFT: 328px; POSITION: absolute; TOP: 304px" runat="server"
                Width="176px" Height="24px" Font-
Bold="True">Level</asp:label><asp:textbox id="Textbox2" style="Z-INDEX:
106; LEFT: 536px; POSITION: absolute; TOP: 296px"
                runat="server" Width="144px"
Height="36px"></asp:textbox><asp:label id="Label2" style="Z-INDEX: 100;
LEFT: 328px; POSITION: absolute; TOP: 248px" runat="server"
                Width="176px" Height="24px" Font-
Bold="True">Temperature</asp:label><asp:label id="Label6" style="Z-INDEX:
105; LEFT: 408px; POSITION: absolute; TOP: 168px" runat="server"
                Width="168px" Height="24px" Font-Bold="True" Font-
Size="Medium" ForeColor="Blue">RECEIVED VALUE</asp:label><asp:label
id="Label1" style="Z-INDEX: 102; LEFT: 328px; POSITION: absolute; TOP:
248px" runat="server"
                Width="176px" Height="24px" Font-
Bold="True">Temperature</asp:label><asp:label id="Label3" style="Z-INDEX:
104; LEFT: 240px; POSITION: absolute; TOP: 24px" runat="server"
                Width="560px" Height="32px" Font-Bold="True" Font-
Size="Large" ForeColor="Red">REAL TIME WEB BASED MULTI PURPOSE
WSN</asp:label></form>
        <script runat="server">
    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs)
  'Put user code to initialize the page here

            Dim ts, tss As String
```

```
                Dim con As SqlConnection = New
SqlConnection("server=wsn.computertricks.net;uid=wsn;password=cheran;databas
e=computer")
                Dim com As SqlCommand
                Dim adp As SqlDataAdapter
                Dim ds As DataSet
                Dim sql As String

                sql = "select * from receive"
                com = New SqlCommand(sql, con)
                adp = New SqlDataAdapter(com)
                ds = New DataSet
                adp.Fill(ds, "receive")
                ts = ds.Tables(0).Rows(0)(0)
                tss = ds.Tables(0).Rows(0)(1)
                TextBox1.Text = Mid(ts, 2)
                TextBox2.Text = Mid(tss, 2)
            End Sub
            </script>
        </body>
</HTML>
```

REFERENCES

- Simultaneous web-based real-time temperature monitoring using multiple wireless sensor networks 2005 *IEEE. Jer Hayes, Karl Crowley and Dermot Diamond.*
- Peter A. Blume: *The LabVIEW Style Book*, February 27 2007, Prentice Hall. Part of the National Instruments Virtual Instrumentation Series series.
- Stephen Walther: *ASP.NET 3.5 Unleashed*, December 28 2007, Sams Publishing
- Roy Choudhry (2001), "Linear Integrated Circuits" -2nd edition
- www.microchip.com
- www.ni.com
- www.ecircuits.com