# KTechLab

George John
george@space-kerala.org
www.space-kerala.org

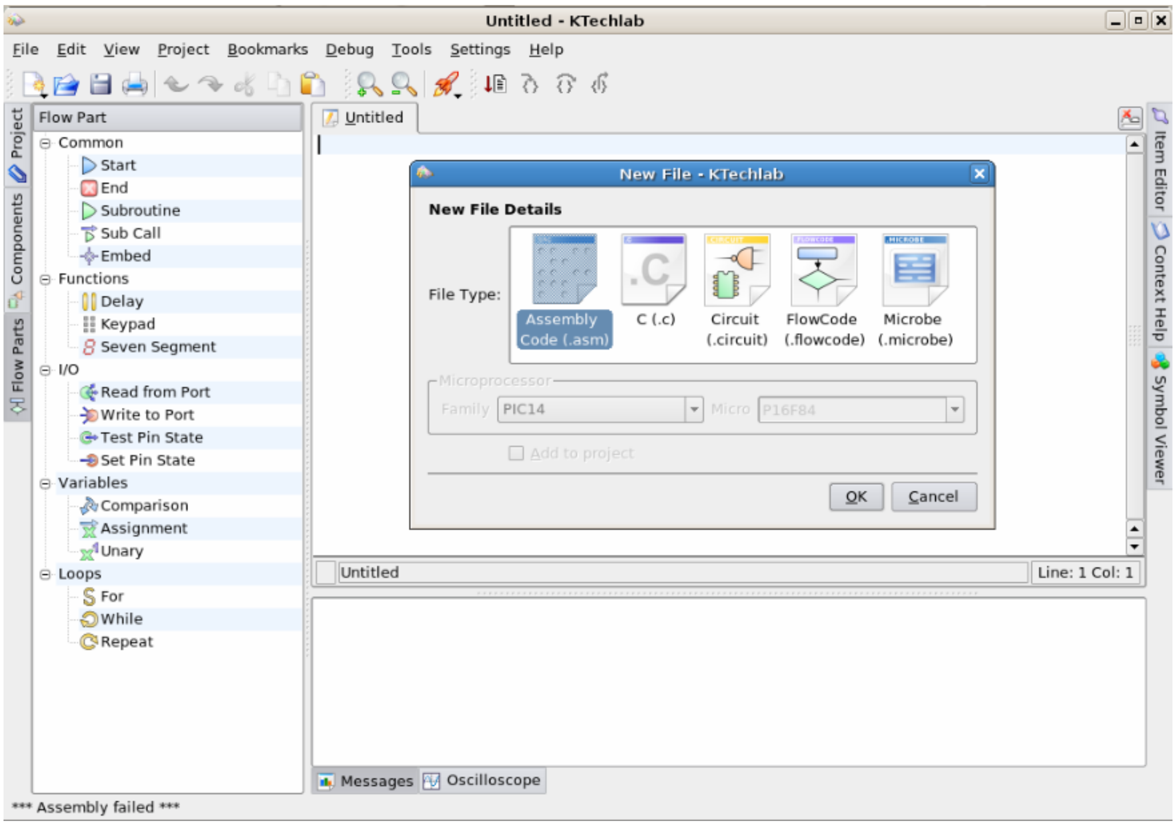
11 June,2007

# Contents

Chapter 1

# Introduction

KTechLab is an Open Source Intergated Design Environment (IDE) for electronic and PIC microcontroller circuit design and simulation. Featuring an extensive circuit designer with autorouting and simulation of many common electronic components and logic elements, KTechLab is the idea tool for educational or hobbyist use. A new file is created either by clicking the new file option in the tool bar or by pressing *Ctrl+n*. Then you will get a window as shown in the below figure.



We get five options,

- Assemble Code
  This option enables us to write the program in *assembly* language. Assembly language is a low-level language. A utility program called assembler, is used to translate assembly language statements into machine code(hex code).
- C Programming
  We can program the microcontroller using C language by choosing this option.C code is compiled using Small Device C Compiler (SDCC).
- Circuit
  Simulation of circuit is done under this section.
- Flow Code
  A flowchart is probably the best place to start microcontroller programming.We can program the microcontroller(PIC) using flowchart.
- Microbe
  Microbe is a programming language like BASIC.

KTechLab supports a wide range of Open Source PIC programmers, enabling finalised PIC programs to be quickly and easily transfered to a real PIC microcontroller.

Chapter 2

# Circuit Simulations

Let us first discuss about some simple circuit simulation. To draw a circuit choose the new file option as *Circuit*. We can choose the components form the Components sidebar.If the components are not avaliable click the *Component* icon in the left sidebar. Components are divided into *seven* categories(0.3.6),

- Sources
  Includes voltage and current sources.
- Discrete
  Includes resistors,capacitors,diodes,transistors etc.
- Switches
  Includes different types of switches
- Outputs
  Includes CRO probes,Voltmeter,Ammeter,LED,Sevensegment display,Matrix display etc.
- Logic
  Includes logical gates such as AND,OR,NOT,XOR,NXOR,Clock etc.
- Connections
  Includes Connection Bus,Serial and Parallel Ports etc
- Intergerated Circuits
  Includes different types of IC'S including PIC.

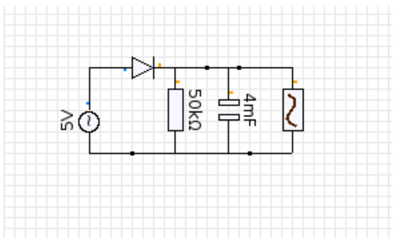
Click the Components you want and drag it into the file. To connect different components first click on the free end of the component. Now the mouse pointer will change to a plus sign. Move the mouse pointer to the component to which, the connection is required. The line become red in color. Now left click the mouse to complete the connection. In order to change values of the component. Just double click the component, to get a combo box in the tool bar. Change the values by using the up and down arrows. In order to see the output waveform, connect the CRO probe Voltage Probe to the Output pins and click the option *Oscilloscope* Oscilloscope in the lower menubar.

## 2.1 Analog Circuits

We can simulate analog circuits using KTechLab.

### 2.1.1 Half Wave Rectifier

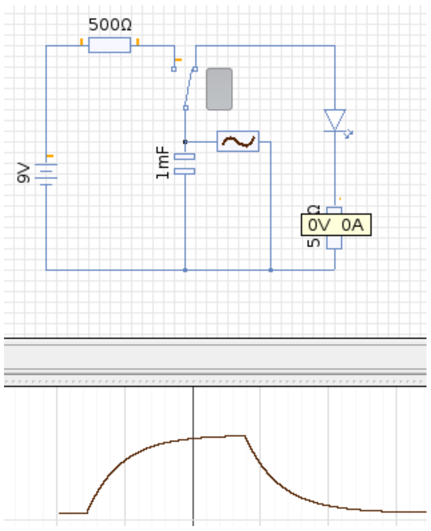
As a first experiment, obtain the wave form of a half wave rectifier.First select an a.c source from the Components sidebar(section SOURCES). Select diode, resistance and capacitor from section *Discrete* and CRO from the *Outputs*. Make connections as shown below.



The output wave form is obtained in the CRO and it is displayed in the window below.

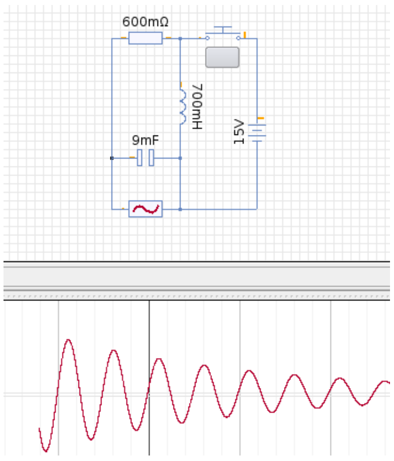
### 2.1.2 Characteristic of a Capacitor

Now study the Characteristic of a Capacitor Using KTechLab. First make the circuit as shown in the below figure. Enable CRO, then charge and discharge the capacitor by using the switch. Observe the wave form. Change the capacitance and study the different curves obtained.



### 2.1.3 Oscillator

Now we see how oscillations are produced in a tank circuit, connection diagram is shown below.

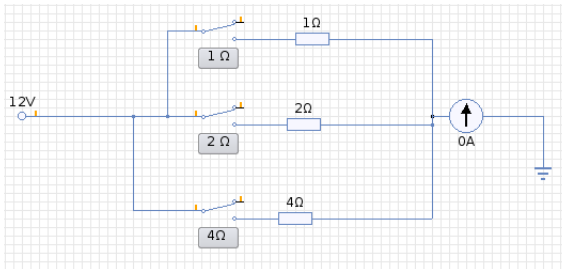


Charge the capacitor by pressing the switch and releasing it. Once the capacitor is charged, one plate of the capacitor has more electrons than the other. When it is discharged through a wire, the electrons return to the postive plate, thus making the capacitor's plates neutral, or discharged. However, this action works differently when you discharge a capacitor through a coil. When current is applied through a coil, a magnetic field is generated around the coil. This magnetic field generates a voltage across the coil that opposes the direction of electron flow. Because of this, the capacitor does not discharge right away. The smaller the coil faster the capacitor discharge. Now the interesting part happens. Once the capacitor is fully discharged through the coil, the magnetic field starts to collapse around the coil. The voltage induced from the collapsing magnetic field recharges the capacitor oppositely. Then the capacitor begins to discharge through the coil again, generating a magnetic field. This process continues until the capacitor is completely discharged due to resistance.

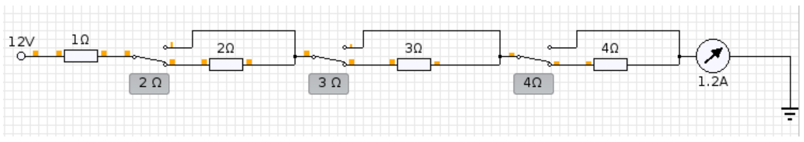
### 2.1.4 Resistor

Find the current through the resistance when connected in parallel and serial mode.Connection diagram is given below.

**Parallel Connection**



**Serial Connection**



## 2.2 Digital Circuits

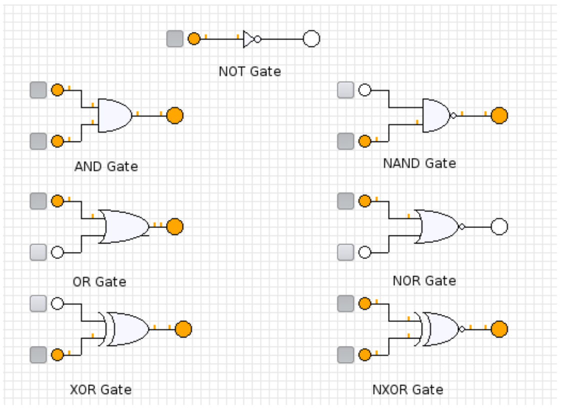
We can Simulate logical circuits using ktechlab. Let us first stimulate logic gates.

### 2.2.1 Logic Gates

Logic gates process signals which represent true or false. Normally the positive supply voltage +Vs represents true and 0V represents false. Some basic gates are discussed below.

- NOT Gate
  The output Q is true when the input A is NOT true
- AND Gate
  if *input A AND input B* are both *true* then the *output* is true.
- OR Gate
  The output Q is true if input A OR input B is true (or both of them are true)
- NAND Gate
  If a NOT gate is connected to the output of AND gate we get a NAND gate
- NOR Gate
  If a NOT gate is connected to the output of OR gate we get a NOR gate

Figure below shows the simulations of Logical gates using KTechLab



### 2.2.2 Flip-flop

Flip-flops are synchronous bistable storage devices capable of storing one bit. In this case synchronous means, the output changes are synchronised with the clock signal, ie the output changes when the clock changes. There are several types of flip-flops, the common ones of which are described in the following paragraphs.

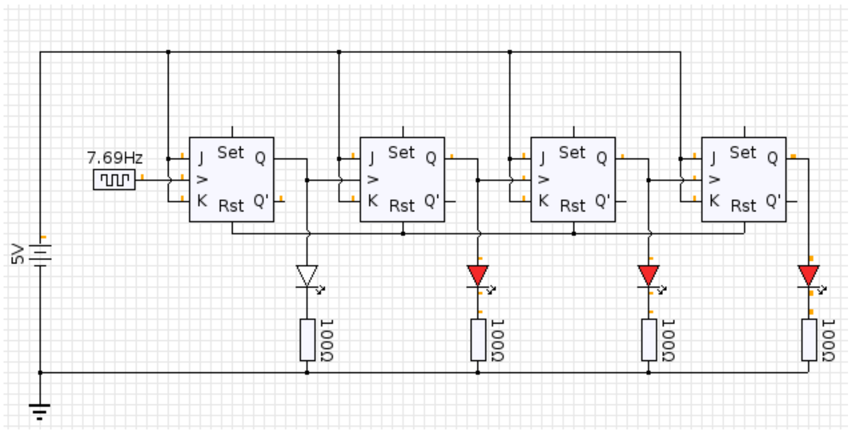
### The Set-Reset (S-R) Flip-flop

The Set-Reset (SR) flip-flop has two inputs, namely, a Set input, or S, and a Reset input, or R. It also has two outputs, the main output Q and its complement Q. A simple representation of an S-R flip-flop is a pair of cross-coupled NOR gates, i.e, the output of one gate is tied to one of the two inputs of the other gate and vice versa. The free input of one NOR gate is used as R while the free input of the other gate is used as S. The output of the gate with the 'R' input is used as the Q output while the output of the gate with the 'S' input is used as the Q output. Thus, resetting an S-R flip-flop's output Q to '0' requires R=1 and S=0, while setting Q to '1' requires S=1 and R=0. In real-world applications, flip-flops are 'clocked' so that one can control the exact moment at which the output changes its state in response to changes in inputs. The clock digital input of clocked flip-flops is usually denoted as C.

### The J-K Flip-flop

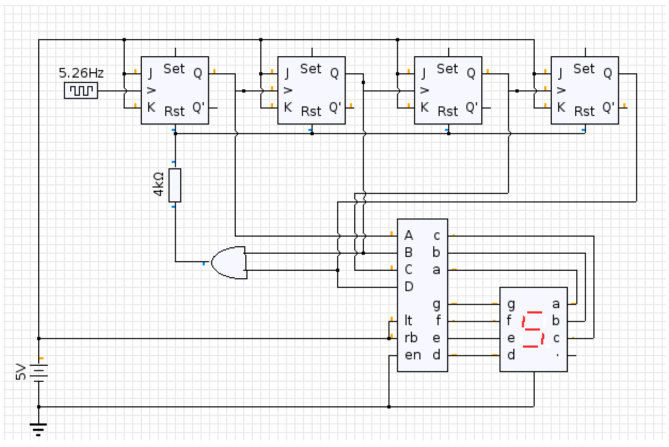
The J-K flip-flop differs from the S-R flip-flop in the sense that its next output is determined by its present output state as well, besides the states of its inputs. Note that in the J-K flip-flop, the S input is now called the J input and the R input is now called the K input. Thus, in a JK flip-flop, the output will not change if both J and K are '0', but will toggle to its complement if both inputs are '1'.

Construct a simple four bit counter using J-K Flip flop. Circuit is shown below. This counter counts from 0000 to 1111, ie is from 0 to 15.



Note that there is a resitance connected between led and the output of the J-K flip flop(Q). This aviods the loading effect.

Let us now construct a Decade counter by modifying the above circuit. The above counter counts from zero to fifteen, we have to *restrict* the counter such that it counts from *zero* to *nine*. For this we connect an AND gate between the Output of second and fourth Filp Flop and the output of the AND gate is connected to the reset pin of all the flip flop. When counter counts ten the output of the second and fourth flip flop become high, and the counter gets reset to 0000.



There are serval IC's given in the Intergerated Circuit section. Some of them are,

### 2.2.3 Intergerated Circuits

Integrated circuits were made possible by experimental discoveries which showed that semiconductor devices could perform the functions of vacuum tubes, and by mid-20th-century technology advancements in semiconductor device fabrication. The integration of large numbers of tiny transistors into a small chip was an enormous improvement over the manual assembly of circuits using discrete electronic components. The integrated circuit's mass production capability, reliability, and building-block approach to circuit design ensured the rapid adoption of standardized IC'S in place of designs using discrete transistors.

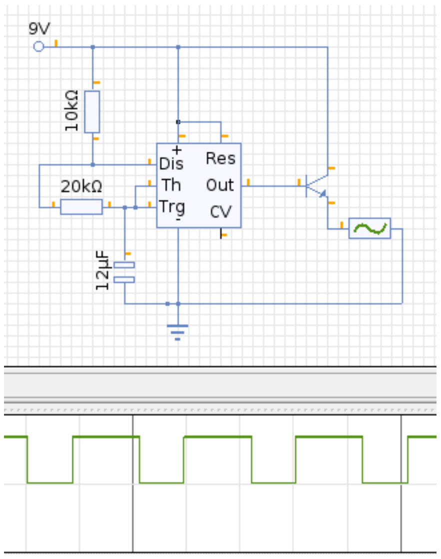
There are two main advantages of ICs over discrete circuits: cost and performance. Cost is low because the chips, with all their components, are printed as a unit by photolithography and not constructed a transistor at a time. Performance is high since the components switch quickly and consume little power, because the components are small and close together. Among the most advanced integrated circuits are the microprocessors or "cores", which control everything from computers to cellular phones to digital microwave ovens. Digital memory chips and ASICs are examples of other families of integrated circuits that are important to the modern information society. In the below section we will discuss about various IC simulations avaliable in the KTechlab.

#### PIC

PIC is a family of Harvard architecture microcontrollers made by Microchip Technology, derived from the PIC1650 originally developed by General Instrument's Microelectronics Division.PICs are popular with developers and hobbyists alike due to their low cost, wide availability, large user base, extensive collection of application notes, availability of low cost or free development tools, and serial programming (and re-programming with flash memory) capability.

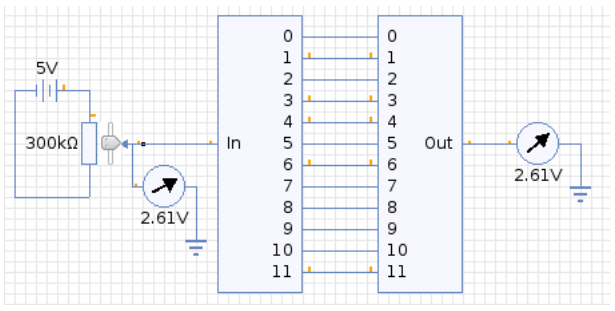
#### 555 Timer

The 555 family of timer chips can be used in either monostable mode or astable mode. In the monostable or "one shot" mode, each time the 555 timer is triggered, the output will go high for a specified amount of time, then return to low and await another trigger signal. In the astable mode, the timer triggers itself periodically and becomes an oscillator, sending out a train of pulses. Figure below shows a square wave generator using 555.



## ADC/DAC

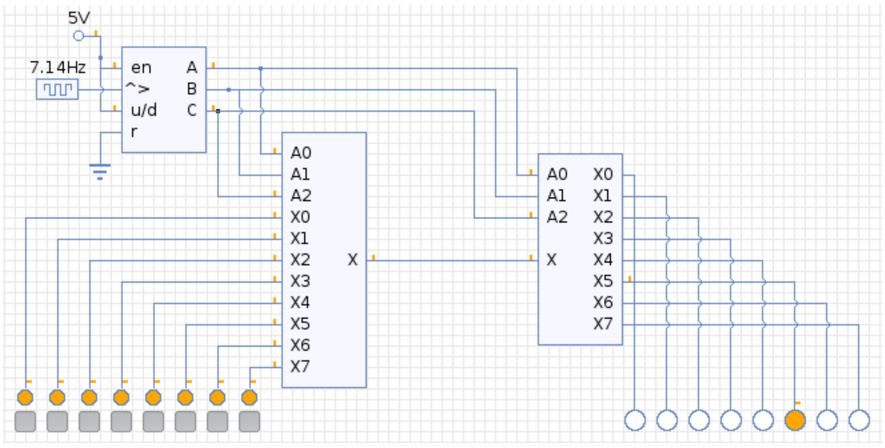
Analog-to-digital converter (abbreviated ADC, A/D or A to D) is an electronic internal circuit (i/c) that converts continuous signals to discrete digital numbers. The reverse operation is performed by a digital-to-analog converter (DAC). Typically, an ADC is an electronic device that converts an input analog voltage ( or current ) to a digital number. The digital output may be using different coding schemes, such as binary and two's complement binary. Figure below shows ADC and DAC together, ADC convert the input voltage to digital and DAC reconvert it into analog. It is seen that the accuracy increases as the number of bit increases(Since resolution increases). Number of bits can be increased by double clicking the ic and changing the value in the combo box.



## Multiplexer and Demultiplexer

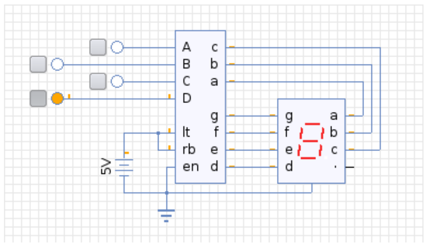
A multiplexer or mux (occasionally the term muldex is also found, for a combination multiplexer-demultiplexer) is a device that selects one of many data-sources and outputs that source into a single channel.

A demultiplexer (or demux) is a device taking a single input that selects one of many data-output-lines and connects the single input to the selected output line. A multiplexer is often used with a complementary demultiplexer on the receiving end.



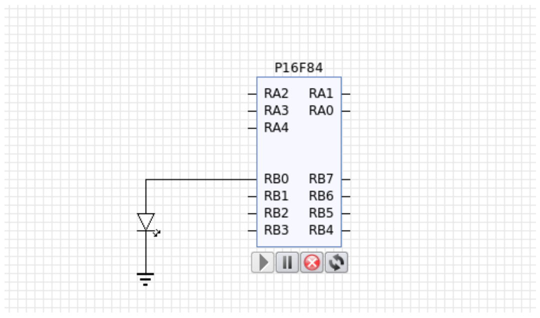
### BCD to Sevensegment Decoder

A BCD to Seven Segment Decoder inputs data in BCD form and converts it to a seven segment output. The IC does the decoding that is involved in activating the appropriate segment outputs (a-g) that is required to represent the binary number that is input. There are 4 binary inputs to the Decoder and seven output segments (a-g).



## 2.3 Microcontroller

Create a new file with circuit as option. Then from the component menu choose the Pic, make proper connections and double click on the pic icon to upload the required program to the Pic . Click the play button on the pic to run the simulation. The figure below shows a sample circuits. Here we upload a program to Blink an led connected to the Port B,0 of the microcontroller with a delay of 100 ms. We will discuss microcontroller programming *later*.



# Chapter 3

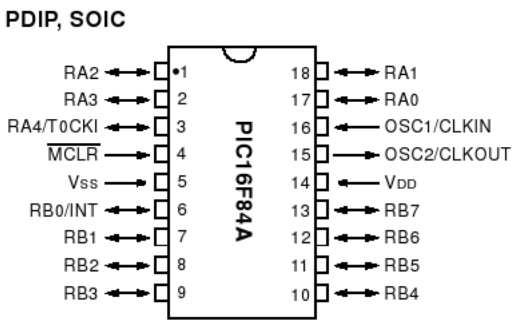
# Microcontroller-16F84

Let have a small discussion about microprocessor. A microprocessor is a programmable digital electronic component that incorporates the functions of a central processing unit (CPU) on a single semiconducting integrated circuit (IC). Microprocessor consists of,

- The CPU core
- Memory (both ROM and RAM)
- Some parallel digital I/O

Microprocessor can execute a stored set of instrucctions to carry out user defined task. Microprocessor when intergerated with timer, serial I/O, ADC we get microcontroller.Microchip manufacture a series of microcontrollers called PIC. Different series of microcontroller are now available in the market. We are going to concentrate on the 16F84 PIC. Once you have learnt how to program one type of PIC, learning the rest is easy.

## 3.1 16F84 Pin Diagram



We will go through each pin, explaining what each is used for,

### 3.1.1 RA0 To RA4

RA is a bidirectional port. That is, it can be configured as an input or an output.The number following RA is the bit number (0 to 4). So, we have one 5-bit directional port where each bit can be configured as Input or Output.

### 3.1.2 RB0 To RB7

RB is a second bidirectional port. It behaves in exactly the same way as RA, except there are 8 - bits involved.

### 3.1.3 VSS And VDD

These are the power supply pins. VDD is the positive supply, and VSS is the negative supply, or 0V. The maximum supply voltage that you can use is 6V, and the minimum is 2V.

### 3.1.4 OSC1/CLK IN And OSC2/CLKOUT

These pins is where we connect an external clock, so that the microcontroller has some kind of timing.

### 3.1.5 MCLR

This pin is used to erase the memory locations inside the PIC (i.e. when we want to re-program it). In normal use it is connected to the positive supply rail.

### 3.1.6 INT

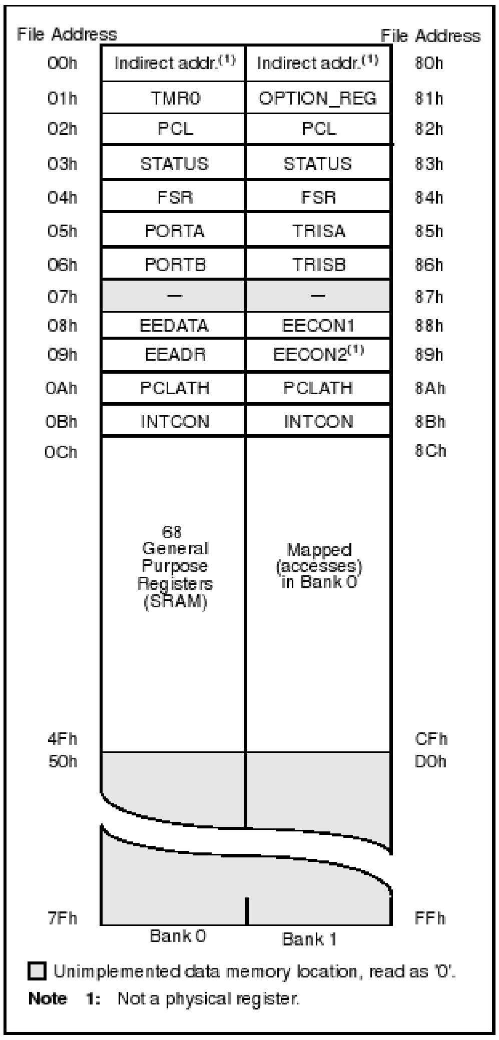
This is an input pin which can be monitored. If the pin goes high, we can cause the program to restart, stop or any other single function we desire. We won't be using this one much.Ex:Interrupt is used in the case of keypad scanning. When one of the key is pressed the interrupt pin will go high, then the microcontrollers will pause the current program, and it will execute a program to determined the key pressed.

### 3.1.7 T0CK1

This is another clock input, which operates an internal timer. It operates in isolation to the main clock. Again, we won't be using this one much either.

## 3.2 The Registers

A register is a place inside the PIC that can be written to, read from or both. Think of a register as a piece of paper where you can look at and write information on.The figure below shows the register file map inside the PIC16F84.



First thing you will notice is that it is split into two - Bank 0 and Bank 1. Bank 1 is used to control the actual operation of the PIC, for example to tell the PIC which bits of Port A are input and which are output. Bank 0 is used to manipulate the data. An example is as follows: Let us say we want to make one bit on Port A high. First we need to go to Bank 1 to set the particular bit, or pin, on Port A as an output. We then come back to Bank 0 and send a logic 1 (bit 1) to that pin.

The most common registers in Bank 1 we are going to use are STATUS, TRISA and TRISB. The first allows us to come back to Bank 0, TRISA allows us to select which pins on Port A are output and which are input, TRISB allows us to select which pins on Port B are output and which are input. The SELECT register in Bank 0 allows us to switch to Bank 1. Let us take a closer look at these three registers,

### 3.2.1 STATUS

To change from Bank 0 to Bank 1 we tell the STAUS register. We do this by setting bit 5 of the STATUS register to 1. To switch back to Bank 0, we set bit 5 of the STATUS register to 0.

### 3.2.2 TRISA and TRISB

These are located at addresses 85h and 86h respectively. To program a pin to be an output or an input, we simply send a 0 or a 1 to the relevant bit in the register. Now, this can either be done in binary, or hex. I personally use both, as the binary does help visualize the port. If you are not conversant with converting from binary to hex and vice versa, then use a scientific calculator.
So, on Port A we have 5 pins, and hence 5 bits. If I wanted to set one of the pins to input, I send a 1 to the relevant bit If I wanted to set one of the pins to an output, I set the relevant bit to 0. The bits are arranges in exactly the same way as the pins, in other words bit 0 is RA0, bit 1 is RA1, bit 2 is RA2 and so on. Lets take an example. If I wanted to set RA0, RA3 and RA4 as outputs, and RA1 and RA2 as inputs, I send this: 00110 (06h). Note that bit zero is on the right, as shown:

| PORTA Pins | RA4 | RA3 | RA2 | RA1 | RA0 |
|---|---|---|---|---|---|
| Bit Number | 4 | 3 | 2 | 1 | 0 |
| Binary | 0 | 0 | 1 | 1 | 1 |

The same goes for TRISB

| PORTB Pins | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 |
|---|---|---|---|---|---|---|---|---|
| Bit Number | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

### 3.2.3 PORTA and PORTB

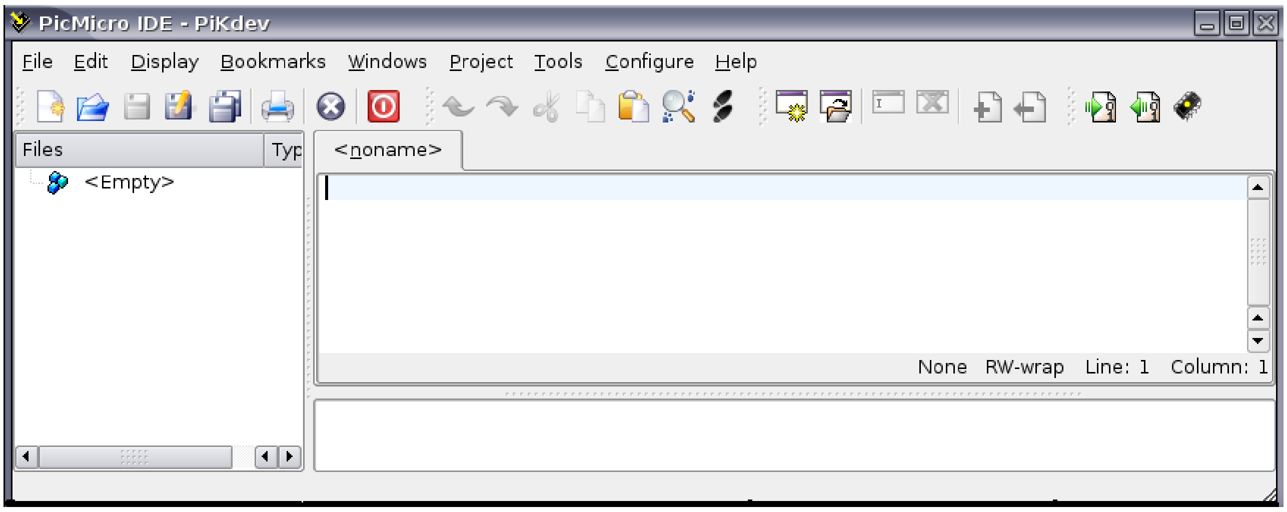
To send one of our output pins high, we simply send a 1 to the corresponding bit in our PORTA or PORTB register. The same format follows as for the TRISA and TRISB registers. To read if a pin is high or low on our port pins, we can perform a check to see if the particular corresponding bit is set to high (1) or set to low (0). Before I give an example code, I need to explain just two more register w and f.

### 3.2.4 W

The W register is a general register in which you can put any value that you wish. Once you have assigned a value to W, you can add it to another value, or move it. If you assign another value to W, its contents are overwritten.

## 3.3 Burning Program to the Chip Using Pikdev

Convert the program in hex by clicking the rocket icon and save it. Burn the program to the chip using Pikdev(To run Pikdev press *Alt+F2* and type *pikdev*).Screenshot of Pikdev is shown below.

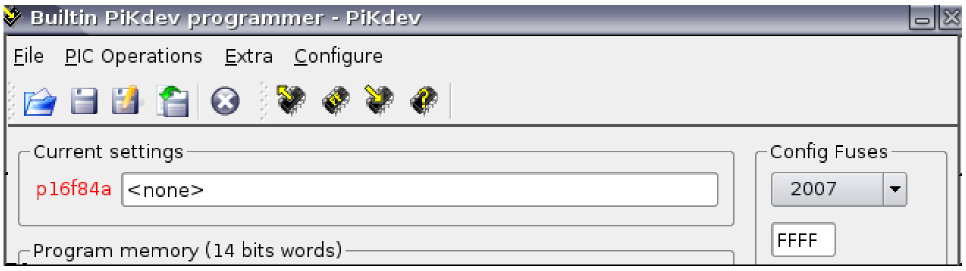

PicMicro IDE - PiKdev
File Edit Display Bookmarks Windows Project Tools Configure Help
Files
Typ
<Empty>
<noname>
None RW-wrap Line: 1 Column: 1

### 3.3.1 Configuring PiKdev

Select Configure menu → Pikdev (or press *Ctrl+Shift+P*). Assembler option.From the window that appears choose the the processor for both assembler and disassembler,in our case the processor selected is p16f84a.Click *Apply* button and press *OK*.

### 3.3.2 Burning Program

To burn the program to the PIC, first click the chip icon in the menu . You will get a new window with four icons as shown below.


Builtin PiKdev programmer - PiKdev
File PIC Operations Extra Configure

Current settings
p16f84a <none>

Config Fuses
2007
FFFF

Program memory (14 bits words)

⇒ Read the data from the chip.

⇒ Erase the program in the chip.

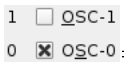
⇒ Write the program onto the chip.

⇒ Verify the program on the chip.

#### Config Fuses

2 ☐ WDT ⇒ **Watchdog timer** is a mechanism which microcontroller uses to defend itself against programs getting stuck. As with any other electrical circuit, so with a microcontroller too can occur failure,

or some work impairment. Unfortunately, microcontroller also has program where problems can occur as well. When this happens, microcontroller will stop working and will remain in that state until someone resets it. Because of this, watchdog mechanism has been introduced. After a certain period of time, watchdog resets the microcontroller (microcontroller in fact resets itself). Watchdog works on a simple principle: if timer overflow occurs, microcontroller is reset, and it starts executing a program all over again. In this way, reset will occur in case of both correct and incorrect functioning. For preventing reset in case of correct function reset the watchdog timer.


1 ☐ OSC-1
0 ☒ OSC-0

⇒**Selecting Oscillator**

There are several basic methods to provide a clock for the microcontroller. The microcontroller has an internal oscillator and two pins are provided on the 16F84 ( pins 15 and 16 ) to connect the frequency standard for the clock.The 16F84A version is capable of operation from DC to 20 MHz.
The least accurate method would be an RC ( resistor capacitor ) network connected to pin 15. The RC mode can be used when timing is not a critical concern such as when two inputs are high, the desired output goes high. The time constant of the RC network determines the frequency, where
t = R ∗ C
and
f=1 / T
It is recommended that if you do plan to use the RC frequency standard method, the resistor values should remain between 5 K and 100 K which would require a capacitor value of approximately 20 pF. Resistances $\leq$ 2.2k may cause the oscillator to become unstable or stop altogether. Extremely high resistances such as 1 Meg can cause the oscillator to become overly sensitive to noise, humidity and leakage.
Quartz crystals provide the highest accuracy. Along with the crystal, two capacitors of approximately 22 pF will be required (see the listing below).Microchip recommends the following capacitor values for crystal oscillators. Higher capacitance will increase the stability of the oscillator but will also increase the start-up time.

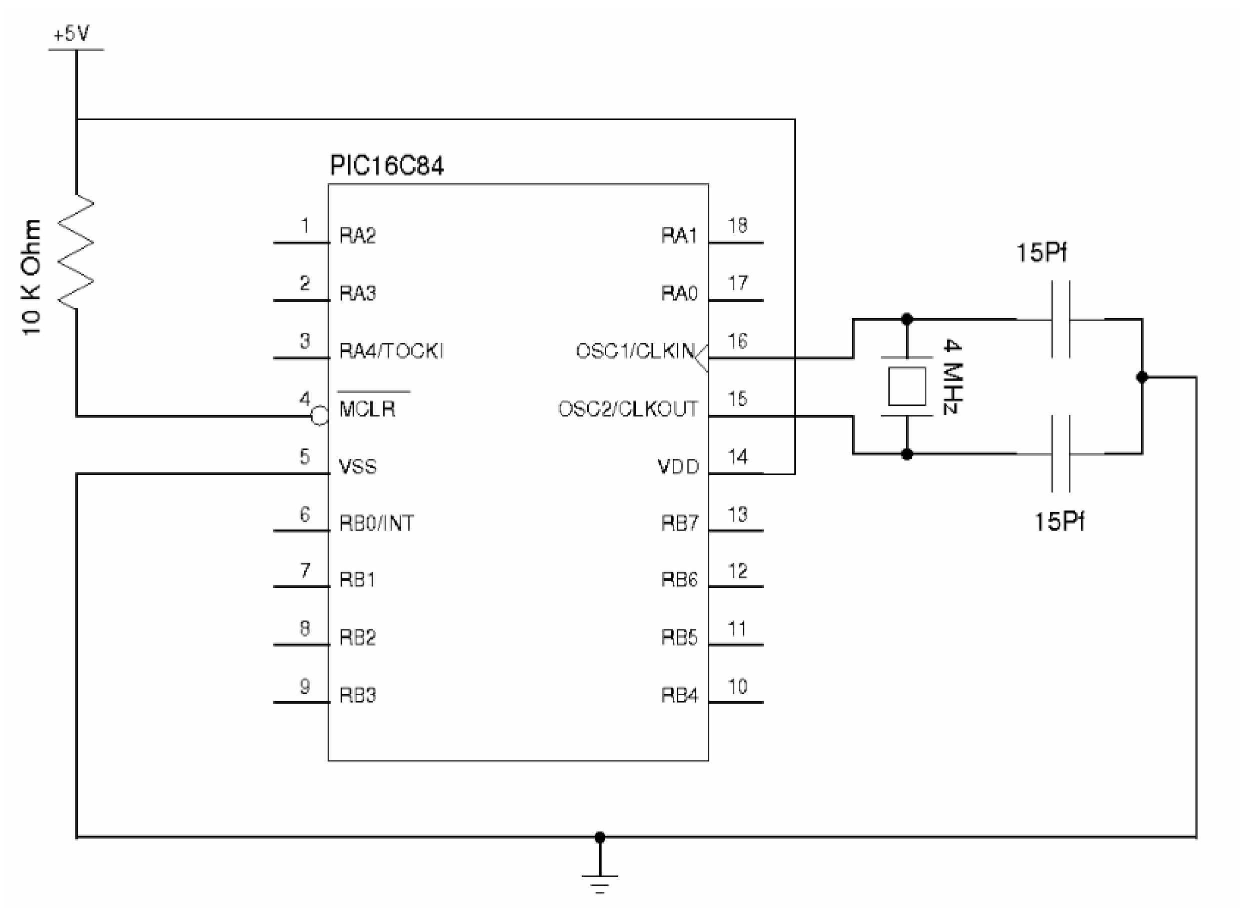
| Mode | Frequency | Capacitance |
|---|---|---|
| LP | 23 kHz | 68 100 pF |
| LP | 23 kHz | 68 100 pF |
| LP | 200 kHz | 15 33 pF |
| XT | 100 kHz | 100 150 pF |
| XT | 2 MHz | 15 33 pF |
| XT | 4 MHz | 15 33 pF |
| HS | 4 MHz | 15 33 pF |
| HS | 10 MHz | 15 33 pF |

RC LP XP or HS option is selected by setting or reseting OSC-0 and OSC-1 in the config-Fuses

| Mode | OSC-1 | OSC-0 |
|---|---|---|
| LP | 0 | 0 |
| XT | 0 | 1 |
| HS | 1 | 0 |
| RC | 1 | 1 |

First erase the data in the chip by clicking the *Erase button*.Choose the hex file to be burned and click the *Programming button* .

## 3.4 Basic Circuit Diagram of a PIC Micro Controller



The above figure shows the basic circuit diagram of a PIC microcontroller(for any Pic series). A 4Mhz crystal is connected to the pin OSC 1 and OSC 2. MCLR is connected to the +5 volt supply through a 10 K Ohm resistance.

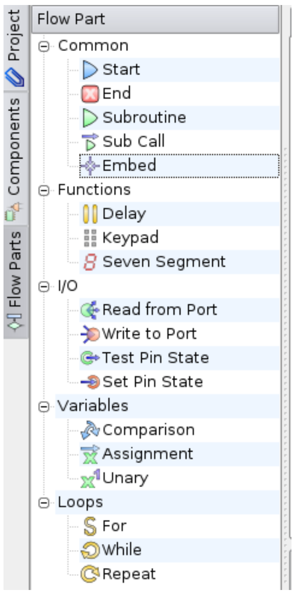
Chapter 4

# Flow Code

Let us now discuss programming via flowcode. Click on the newfile icon and choose the flowcode option from the submenu. You will now see a picture of the PIC chip on the top left of the window, this show its initial settings(it is not part of the flowchart). You can change these initial settings by clicking on each connects of the PIC to turn it on or off. Clicking and dragging left or right will set it to be an input or an output. Flowcharts need a start and end which you can drag in from the flow part sidebar. You can then drag in any of the other flowchart actions and connect them together.

## 4.1 Flowcode Symbols

Flowcode symbols are similar to flowchart symbols. The Flow part window consists of five sections,



- Common
  Contains *Start,Stop,Subroutine,SubCall and Embed*

Start ⇒ Beginning of the program.
End ⇒ End of the program.
Subroutine ⇒ Allow the programmer to separate the program into smaller parts which are easier to solve. A subroutine can be called many times in the same program, thus avoid the repetition of code.
Sub Call ⇒ Call subroutine.

- Functions
  Contains *Delay,keypad and Sevensegment*
  Delay ⇒ Delay the process for a specified time.
  Keypad ⇒ Read data from the keypad.In order to use this, first click the advanced icon in the diagram of the pic and create pin map definitions.
  Seven Segment ⇒ Display the number in a sevensegment.This function convert the number into bcd format.

- I/O
  Contains *Read from Port,Write to Port,Test pin state and Set pin state*
  Read from Port ⇒ Read data from port to vairable.
  Write to Port ⇒ Write data to the port.
  Test Pin State ⇒ Used to test the state of a particular *pin.*
  Set Pin State ⇒ For setting the state of a particular pin.

- Variables
  Contains *Comparison,Assignment and Unary*
  Comparison ⇒ Compare variables.
  Assignment ⇒ Assign values to a variable.
  Unary ⇒ Performs unary operations such as rotate left, rotate, right, increment and decrement.

- Loops
  Contains *For,While and Repeat*
  Loops can be created to execute a block of code for a fixed number of times or (e.g. 7 or 17 or 70). Alternatively, loops can be created to repetitively execute a block of code until a condition changes to a wanted state. For instance, the loop may continue until a condition changes from false to true, or from true to false. In this case, the block of code being executed must update the condition being tested in order for the loop to terminate at some point. If the test condition does not change somehow within the loop, the loop will never terminate.(endless loop). This occurs due to a logical error.

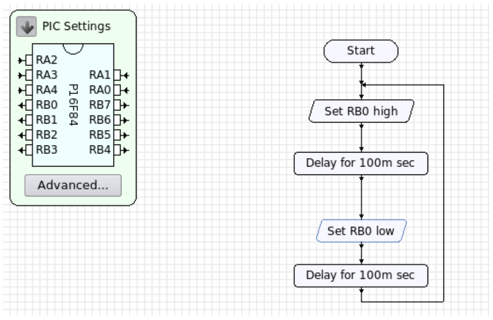
Dragging in a subroutine adds a large diagram item into which you can put other items. I am restricting our discussion to PIC 16F84A Microcontroller in the comming sections

## 4.2 Writing to the I/O ports

Microcontroller Port's can be configure as Input or Output.We can write data to an ouput port. Draw a flowcode to set and reset the PORTB,0 pin with an interval of 100 millisec. First we have to set the PORTB,0 pin as output. This can be done in two ways.

∗ Click the arrow on the PORTB,0 pin an drag it to left

∗ Click advanced.... and set Port B,0 pin as output by re-setting the last bit of tris register of Port B. Draw the flowcode and save it.

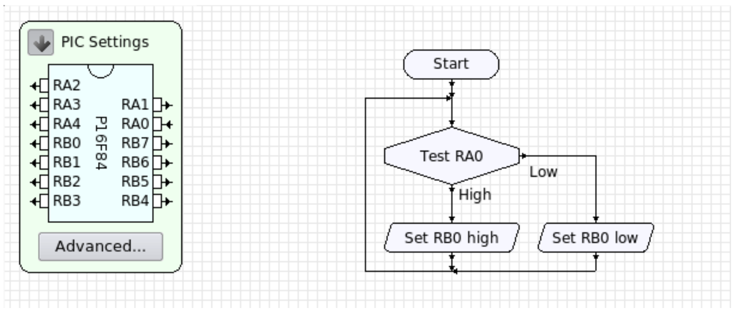


You can see that there is no end statement in the flowcode. This is to make the code run indefinitely. If you insert an end in the flowcode,the code will execute only once and we are not able to see any output. TO simulate the flowcode, follow the procedure as described in the section *Simulation-Microcontroller.* Setting and resetting PORTB,0 is equivalent to writting 0 and 1 to PORTB. Since PORTB is *eight* bit we can write 0 to 255 to PORTB. If we write eight to PORTB the RB3 of PORTB will be set. Look at the table below.

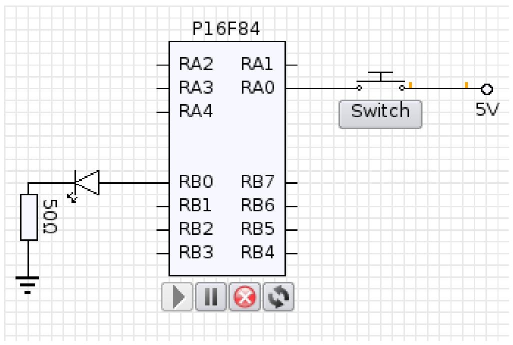
| PORTB Pins | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 |
|---|---|---|---|---|---|---|---|---|
| Bit Number | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Binary | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

## 4.3 Reading from the I/O ports

We can read data from the port when it is configured as input. Draw a flowcode to test the pin state of PORTA,0 and set set PORTB,0 if high otherwise reset,
Since by default ports are configured as inputs, there is no need to configure PORTA,0 pin as input. Configure zero-th pin PORTB as output by clicking the arrow on the PORTB,0 pin an drag it to left. Flowcode is given below,



Save the flowcode and simulate it. Circuit for simulating the above code is shown below.



## 4.4 More Programs

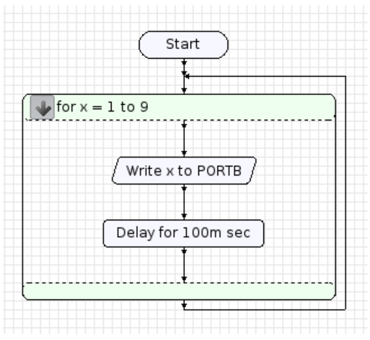
**Draw a flowcode to display the binary equilavent of numbers from Zero to Nine**

Unlike the decimal system, only two digits *0, 1* suffice to represent a number in the binary system. The binary system plays a crucial role in computer science and technology. The first 20 numbers in the binary notation are : 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, 10000, 10001, 10010, 10011 and 10100, the origin of which may be better understood if they are re-written in the following way:

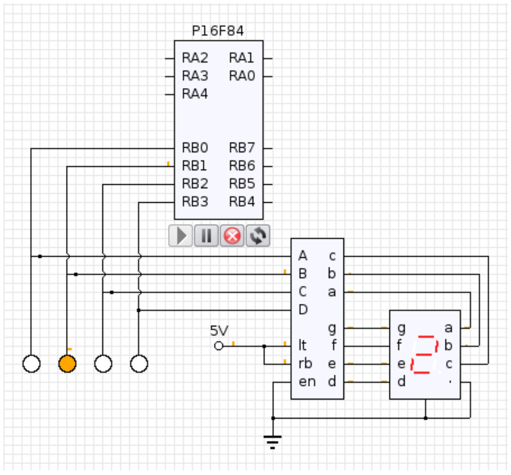
| | | | |
|---|---|---|---|
| 1: | 00001 | 11: | 01011 |
| 2: | 00010 | 12: | 01100 |
| 3: | 00011 | 13: | 01101 |
| 4: | 00100 | 14: | 01110 |
| 5: | 00101 | 15: | 01111 |
| 6: | 00110 | 16: | 10000 |
| 7: | 00111 | 17: | 10001 |
| 8: | 01000 | 18: | 10010 |
| 9: | 01001 | 19: | 10011 |
| 10: | 01010 | 20: | 10100 |

In the case of digital circuit a high voltage level, for example 5v is represented by '1' and low voltage level in most case represented by '0' volt.

First we have to choose the ports: Here i am choosing PORTB. Since this is an eight bit port, the command PORTB=1 is equilavent to writing 00000001 to PORTB, ie the decimal number is converted to binary and then writen to the port.So the maximum possible number which can be writen to the PORTB is 11111111 = 255. Similarly in the case of PORTA(5 bit) the maximum number which can be writen to the port is 11111=31. Set PORTB pin as $OUTPUT$,and draw the flowcode as given below.

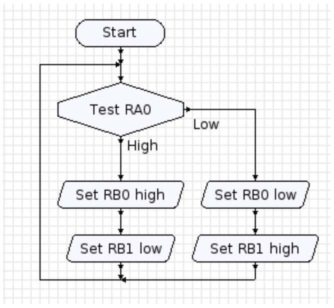


In this case we are using a for loop. The loop will execute until the value of x becomes nine. when x becomes greater than nine then loop will exit. Circuit is given below,

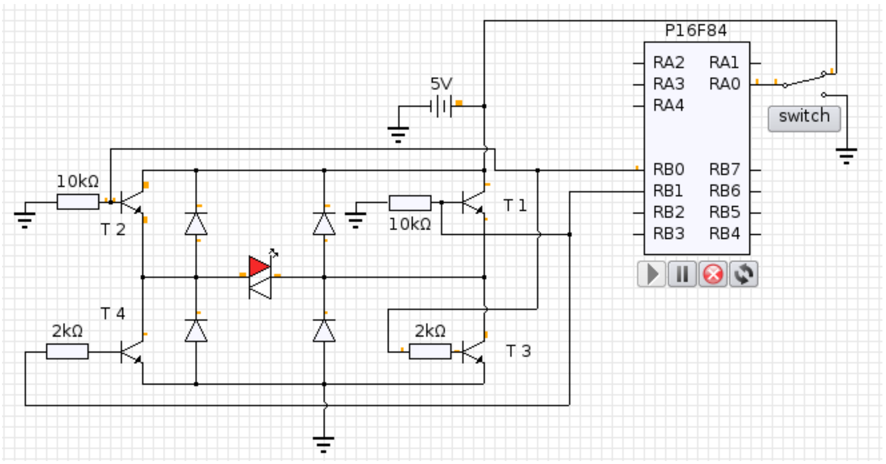


**Program to drive a dc motor to the left or right depending on whether PORTA,0 is high or low.**

A D.C motor can be made to rotate to the left or right by changing the polarity of the d.c supply. Select two pins of PORTB as outputs and PORTA,0 pin as input. When the input is high,set PORTB,0 pin as high and PORTB,1 pin as low and vice versa.



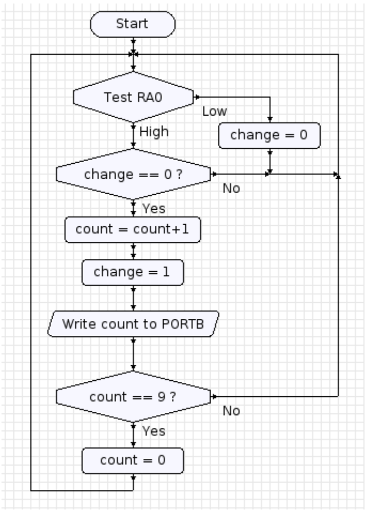
Since the d.c motors consume large current, we have to use a separate transistor circuit to drive the motor. Circuit is given below.



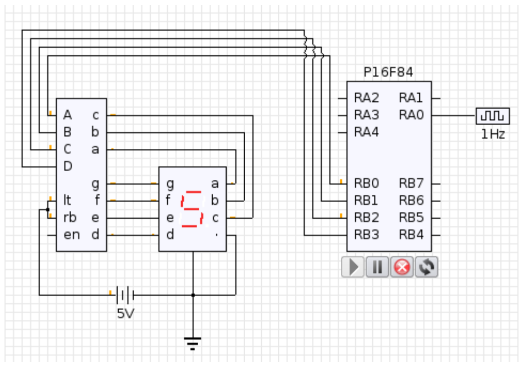
When *RB0* is high transistor *T2* and *T3* conduct; current path is *battery* → *T2* → *motor* → *T1 and gound*. when *RB1* is high transistor *T1* and *T4* conducts; current path is *battery* → *T1* → *motor* →*T4 and gound*. Doide's protect the transistor from back E.M.F

**Design a counter using Microcontroller(counts from 0 to 9)**

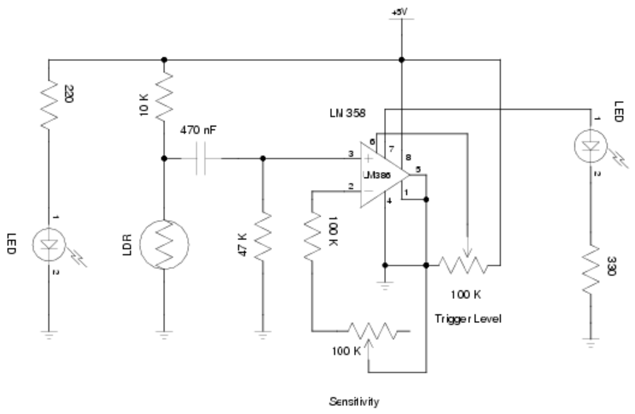
Input PORTA,0 pin is selected as input and PORTB as output.
Program logic: Declare two variables count and change. When RA0 is high count increases by one (*count=count+1*). Otherwise there is no increment. Consider the input source as square wave generator with a frequency of 1 KHz. The output willbe high for 500 millisec but the microcontroller operates at a frequency of 4MHz count will increase more than ones during a single pulse. First we check if change is zero if it is count is increase by one and change is set to one. The variable change become zero only when the clock become low. See the flowcode below.



Circuit for simulation



Practical cases replace the clock and include appropriate circuits, sample circuit below. The led light is cut the output will go high. We can use this circuit for counting number of persons entering the room etc.



Components required IC LM 358, LDR, LED(red), Resistance(100k, 47k, 10k, 330 ohm and 220 ohm), preset(100k-2) and Capacitor(470 nF).

# Chapter 5

# Microbe

Microbe is a simple language which helps to program the microcontroller easily. Presently the microbe supports PIC 16F84, 16F628, 16F627 and 16F877.The language constructs are given below.

## 5.1 Loops

- Goto
  label:
  {
  statement 1
  statement 2
  statement 3
  ..................
  statement n
  }
  goto label

- For
  The for loop can execute a block of code for a fixed or given number of times. Its syntax is as follows.
  for *initialization* to *test conditions*
  {
  block of code
  }
  The above loop will execute until the test condition fails.

- While
  The while loop is used to execute a block of code as long as some condition is true. If the condition is false from the start the block of code is not executed at all. The while loop tests the condition before it's executed so sometimes the loop may never be executed if initially the condition is not

met. Its syntax is as follows.
while condition
{
block of statements executed if condition is true
}

- Repeat
  The do loop also executes a block of code as long as a condition is satisfied. The difference between a ”repeat ...until” loop and a ”while ” loop is that the while loop tests its condition before execution of the contents of the loop begins; the ”repeat” loop tests its condition after it's been executed at least once. As noted above, if the test condition is false as the while loop is entered the block of code is never executed. Since the condition is tested at the bottom of a repeat loop, its block of code is always executed at least once.Its syntax is as follows.
  repeat
  {
  block of statements to be executed until the condition is true
  }
  until condition

## 5.2 Conditional Statements

- If Statement
  *if* condition *then*
  {
  block of statements executed if condition is true.
  }
  *else*
  {
  block of statements executed otherwise.
  }

## 5.3 Example codes

First we have to specify the microcontroller type this is done the statement *P16F84*. Configure PORTs as Input or Output by setting or resetting the bits in the TRIS register.

### 5.3.1 Writting to I/O ports

We have already discussed about writting to I/O ports. Now let us see how this can be done using *microbe*. Write program in microbe to set and reset the PORTB,0 pin with an interval of 100 millisec.
P16F84

```
TRISB = 0
PORTA = 0
PORTB = 0
setpin:
    PORTB.0 = high
    delay 100
    PORTB.0 = low
    delay 100
goto setpin
end
```

In above program we use GOTO statement here the label used is *setpin*
delay 100 means Delay the process for 100 millisecond
PORTB.0 = high means set the PORTB,0 as high.
PORTB.0 = low means set the PORTB,0 as low.

### 5.3.2 Reading from I/O ports

**Program to read the state of PORTA,0 pin and set PORTB,0 if high and vice versa.**
Set PORTA,0 as input , PORTB as output by TRISA=1 and TRISB=0 statements respectively. Clear the PORT'S by using the Statements PORTA=0 and PORTB=0. Test the PORTA,0 pin by *if condition* if high set PORTB,0 else PORTB,0 = low. Program is given below,

```
P16F84
TRISA = 1
TRISB = 0
PORTA = 0
PORTB = 0
testpin:
if PORTA.0 is high then
{
    PORTB.0 = high
}
else
{
    PORTB.0 = low
}
goto testpin
end
```

### 5.3.3 More Programs

**Program to display the binary equilavent of a from Zero to Nine**

Select PORTB as output, program is given below

```
P16F84
TRISB = 0
PORTB = 0
forloop:
for x = 1 to 9
{
    PORTB = x
    delay 100
}
goto forloop
end
```

Note : PORTB = x means write x to the PORTB.

**Program to drive a D.C motor to left or right depending upon whether PORTA,0 is high or low**

```
P16F84
TRISB = 4
PORTB = 0
testpin:
if PORTA.0 is high then
{
    PORTB.0 = high
    PORTB.1 = low
    goto testpin
}
else
{
    PORTB.0 = low
    PORTB.1 = high
    goto testpin
}
end
```

**Design a Counter using microcontroller**

As in the case of flowcode hera also we select RA0 as input pin and PORTB as output port. Code is given below,

```
P16F84
count = 0
x = 0
```

```
TRISB = 0
PORTA = 0
PORTB = 0
testpin:
if PORTA.0 is high then
{
    if x == 0 then
    {
        count = count+1
        x = 1
        PORTB = count
        if count == 8 then
        {
            count = 0
        }
        goto testpin
    }
    else
    {
        goto testpin
    }
}
else
{
    x = 0
    goto testpin
}
end
```

#  Chapter 6

# Small Device C Compiler

SDCC (Small Device C Compiler) is free open source, retargettable, optimizing ANSI-C compiler by Sandeep Dutta designed for 8 bit Microprocessors. The current version targets Intel MCS51 based Microprocessors (8031, 8032, 8051, 8052, etc.), Dallas DS80C390 variants, Freescale (formerly Motorola) HC08 and Zilog Z80 based MCUs. It can be retargeted for other microprocessors, support for Microchip PIC, Atmel AVR is under development. The entire source code for the compiler is distributed under GPL. SDCC uses ASXXXX and ASLINK,an open source retargetable assembler and linker. SDCC has extensive language extensions suitable for utilizing various microcontrollers and underlying hardware effectively.
If you have some basic ideas about C programmming it is easy to program the microcontrolle using SDCC.Let us discuss about a small program to read data from PORTA and output it on PORTB.

```
/∗ Define processor and include header file. ∗/
♯define _16f84
♯include”pic/pic16f84.h”
int x;
void main(void)
{
        TRISA =0x1F;
        TRISB =0x00;
        while(1)
        {
                x=PORTA;
                PORTB = x;
        }
}
```

To compile the program use command *sdcc -mpic14 -p16f84 filename.c.* This will generate asm,hex and many releated code.To simulate this program using ktechlab double click on the pic and upload the program filename.cod.