Embedded System Design (ECE-424E)

Unit 4: Programming with Microcontrollers

Arithmetic operations:-

PIC supports both addition and substraction. Flags C, DC and Z are set depending on a result of addition or substraction, but with one exception: Since substraction is performed like addition of a negative value, C flag is inverse following a substraction. In other words, it is set if operation is possible, and reset if larger number was substracted from a smaller one. Some microcontrollers also support multiplication and division operations.

The following are some of arithmetic options of operations:-

* Multibyte addition and substraction.
* Multibyte multiplication.
* Multibyte division and modulus (remainder).
* Single byte addition, substraction, multiplication and division.

Some Microcontrollers (PIC) have two ADD instructions with two operans and one ADD instruction with 3 operands. These ADD instructions are designed to perform 8-bit additions. The execution result of the ADD instruction will affect all flag bits of the STATUS register.

Vidhu Rawal
Asstt. Professor

following are some of the common operations:-

① - ADD operation — for example —

```
MOV A, #04
ADD A, #20
MOV A, B
MOV B, #
```

② - ADDC operation — for eg.:

```
MOV  A, #04
ADD  A, #20
MOV  Temp, A
 clr A
ADC  A, #0
MOV  Temp+1, A
```

③ - SUBB operation — for eg.:

```
MOV A, #20
ADD A, #04
```

④ - INC instruction — for eg.:

```
INC R0
```

⑤ - DEC instruction — for eg.:

```
DEC R0
```

⑥ - INC DPTR operation — for eg.:

```
INC DPTR
```

⑦ - Multiply operation — for eg.:

```
MOV A, #077h
MOV B, #042h
MUL A, B.
```

⑧ - Division operation — for eg:

```
MOV A, #077h
MOV B, #042h
 DIV AB.
```

Bit Addressing:

The standard bit-wise operations available in 8051 include ANDing an 8-bit value with another value, ORing the bits of value with another and XORing a value with another value. These instructions do not affect any of the PSW bits. In 8051, individual bits can be manipulated as if they were full bytes.

(1)- anl instruction → for eg.: Instruction: anl A, opera

```
mov A, #077
anl A, #100
```

(2)- anl direct, Acc instruction.
for eg: anl Register, A → Instruction.

```
mov A, #77
anl Register, A
```

(3)- anl direct, Const instruction.
for eg: anl Register, Const.

```
anl P0, #1
```

(4)- orl instruction → for eg: Instruction: orl A, operan

```
Mov A, #77
orl A, #100
```

(5)- orl direct, Const instruction

```
orl P0, #1
```

(6)- xrl instruction

```
mov A, #77
xrl A, #100
```

(7)- xrl direct, Acc instruction.

```
mov A, #77
xrl Register, A.
```

The anl c, Bit and orl c, Bit instructions perform logica operations on the Carry flag and another bit with the result being stored in the Carry flag.

Loop Control

Loop Control instruction is used to allow you to count down inside of a loop and execute repeatedly until the count is equal to zero. following is a program that uses the simulator to demonstrate how a set number of loops can be simply accomplished in 8051:

```
: Mainline
org 0
   clr P1.0
   mov R0, #17

Dlay:
: ##### Inside loop code can go here
   djnz R0, Dlay

   Setb P1.0

Loop:
   ajmp Loop          : Loop here forever when finished
```

In this loop, when the djnz instruction is encountered, R0 of the current bank is decremented. If the result of the decrement is not zero, then the branch is taken. In PROG10, R0 was initialized with decimal 17, which means that the djnz will branch 17 times before it will skip the branch and continue on with the program.

Loop instruction allows pretty simple implementation of "for" loops and are pretty easy to understand.

Stack Operations :

There are two stack operations: PUSH and POP. A PUSH puts a parameters onto a stack and increments the stack pointer to the next available stack element. A POP decrements the stack pointer to the previous parameter put on the stack and returns it. Stack data can be pushed or popped anywhere in RAM. Upon power up, the stack pointer is set to address 007h.

In StackLoop, an incrementing value is pushed onto the stack. Pushing this data onto the stack writes over TestVariable after the third PUSH. Following is an application that uses the simulator to demonstrate how the 8051's stack can be used and some pitfalls to watch out for:-

```
: Mainline
org 0
 mov  TestVariable, #055h
 clr  A
 mov  R0, #8
 StackLoop:
  push ACC
  inc  A
  djnz R0, StackLoop
  mov  A. TestVariable
  Loop:
   ajmp Loop
```

To prevent the stack from overwriting data or going beyond the bounds of memory, make sure that, when you specify the data locations, you understand how large the stack can grow to, and make sure there is sufficient space for it.

Subroutines

Calling subroutines in a 8051 microcontroller or any other microcontroller is quite simple with nothing to look out for or worry.
Following is an application that increments a 16-bit value in a subroutine.

Variable declarations: iEQU 0 jEQU 2

```
:Mainline
  org 0
    ajmp Mainline
  Increment:
    inc i
  cjne R0, #0, Inc_Skip
     inc i+1
  Inc_Skip:
     ret
  Mainline:
      mov i,#0F0h
      mov i+1, #012h
       mov j,#32
  ML Loop:
    acall Increment
    djnz j, ML Loop
  Loop:
    ajmp Loop
```

RAM Direct Addressing –

Following program reads and writes to the 128 bytes of scratchpad RAM available at the start of 8051's data space:

Variable declarations:

R0, R1 and R2 used for data/variable storage.

```
Var1  EQU  8
Var2  EQU  9
Var R0 EQU 0

:Mainline
   org 0
   mov R0, #123
     mov  Var1, #12
     mov  Var2, #34
  : Var1 = Var1 : Var2
        mov   A. Var1
        orl   A. Var2
        mov   Var1. A
        mov   Var R0, #Var1
        mov   @ R0, #0
     Loop:
       ajmp Loop
```

The instructions → `mov A, Var1` / `orl A, Var2` / `mov Var1. A`

Show how variables can be used instead of bank registers for processing data.

```
        mov  Var R0, #Var1
        mov  @R0, #0
```

Var R0 is defined to the address 0 which is also the address of bank 0 register R0. When VarR0 is written to the contents of R0 are lost making the next instruction execute incorrectly.

State Machines

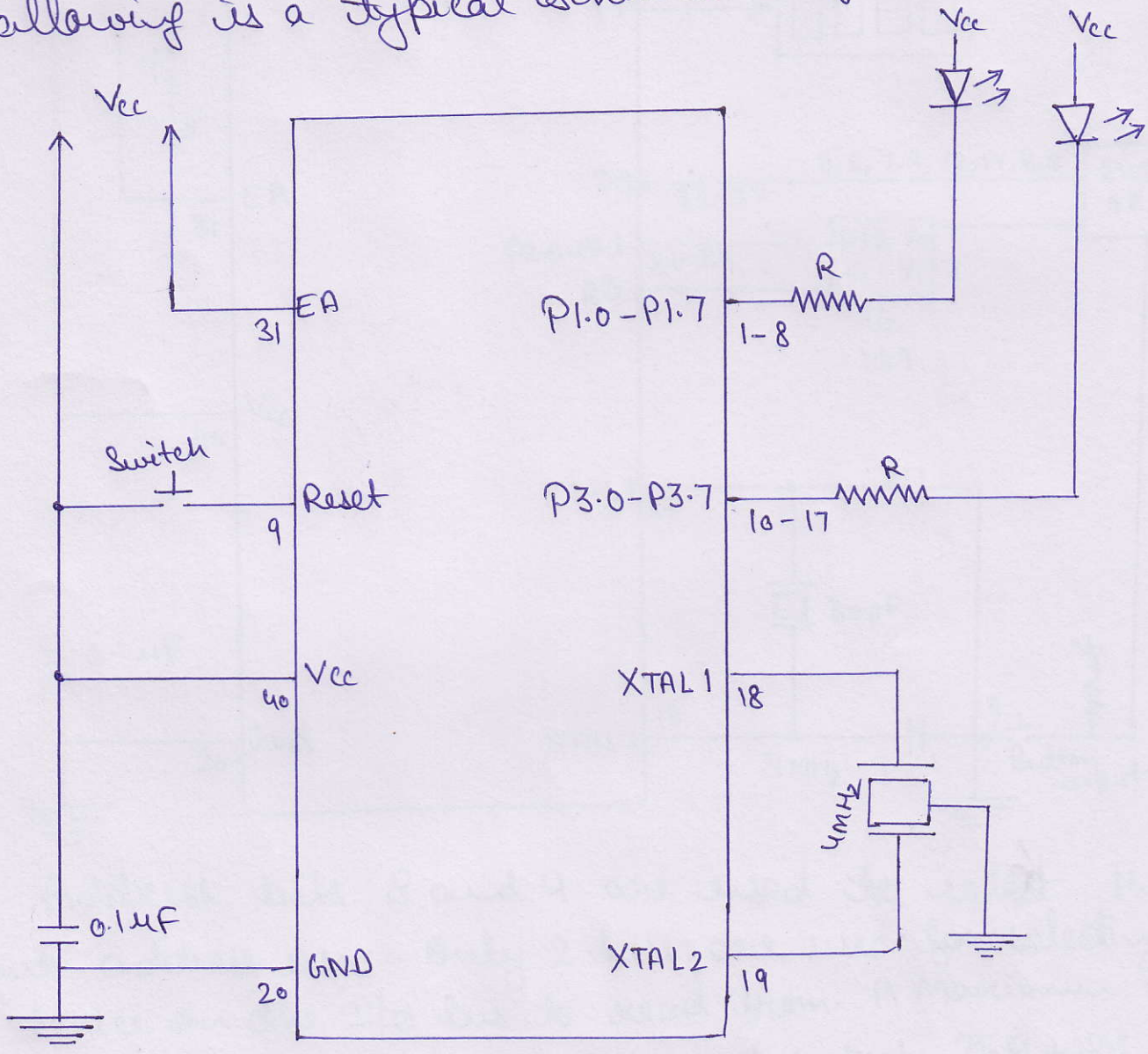
In a state machine, a variable is used to keep track of the current state of the program and help an execution control subroutine decide what to execute next. Following application demonstrates how a set of traffic lights could be controlled in 8051 using a state machine:

```
:Mainline
 org 0
 mov State, #0
  mov P1, 0BBh
  mov DPTR, # Light Table
Loop:
   acall Next light
   Sjmp Loop
Next Light:
   mov A, State
   rl A
   anl A, #0DFh
   jmp @A + DPTR
Light Table:
      Sjmp Greenlight
      Sjmp Yellow light
      Sjmp Red Light
Green Light:
      mov A, #0EBh
      jnb State Dir, GL_EW
       mov A, #0BEh
Yellow Light:
      mov A, #0DBh
       jnb State Dir, YL_EW
        mov A, #0BDh
Red light:
   mov P1, #0BBh
   mov R0, #2
   acall DLay
   mov A, #0F0h
```

Oscillators

Mostly a Crystal is used for providing the timing for the circuits. The crystal circuit is reliable and cheap but is somewhat fragile in high-vibration environments. Another device that is used for the same purpose is Ceramic resonator which is more expensive than crystal and is less accurate but is more robust.

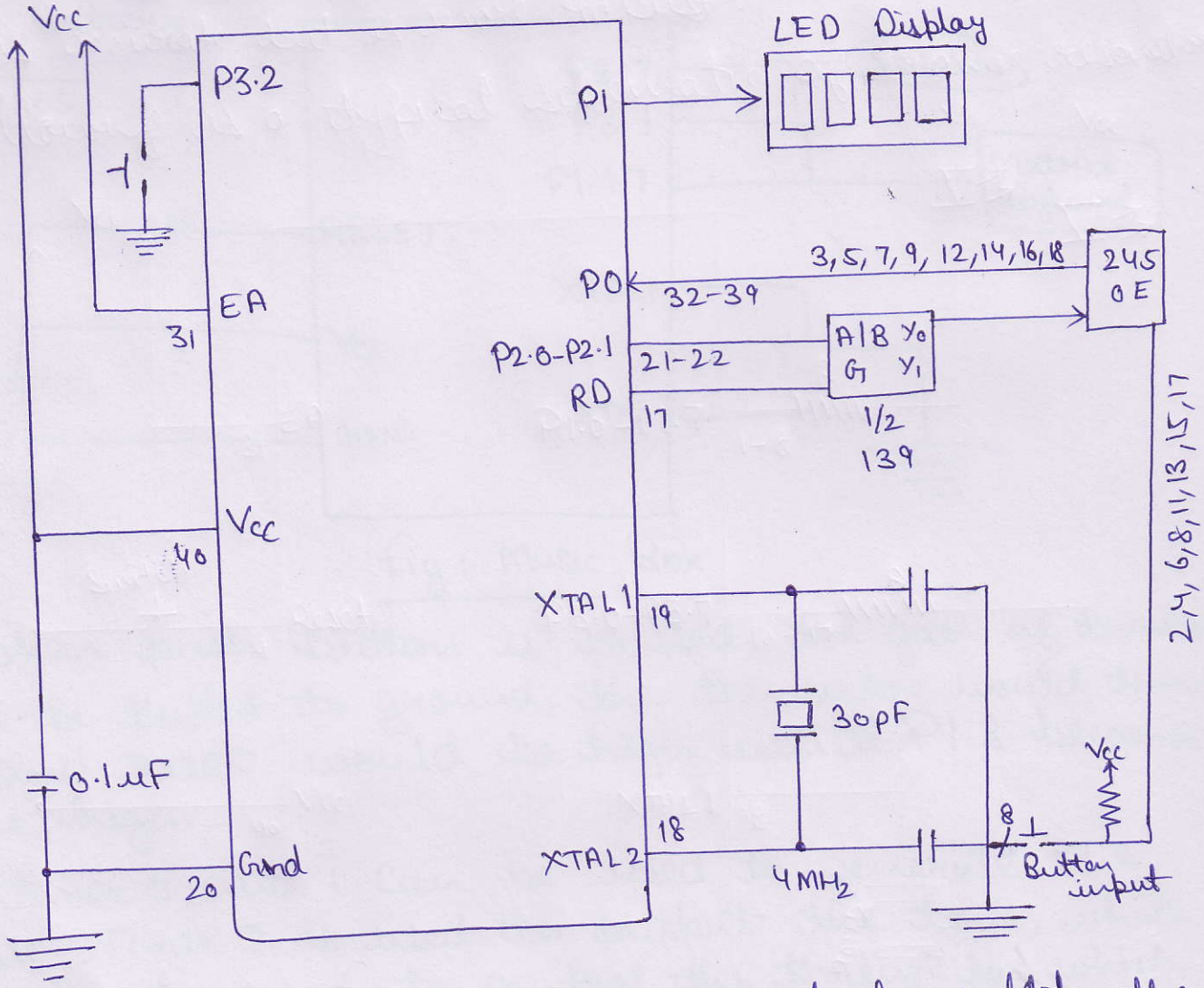
Following is a typical schematic of Ceramic resonator:



Ceramic resonators typically have a 0.5% accuracy which can be used to run the program first with the crystal and then replaced with Ceramic resonator.

Memory-Mapped I/O

Following is a typical circuit for memory-Mapped I/O which uses a button to select between reading addresses 00000h and 00100h. The hardware at these addresses are known as memory-mapped I/O. as is shown in the figure below :-



Address bits 8 and 9 are used to select the input address used. Only 2 bits are used for selecting the device on the I/O bus to read from. A maximum of four I/O devices can be accessed on the bus. This way we can select from software address 00000h, 00100h, 00200h and 00300h.

Music Box :-

The music box is made with the help of 8051 and a speaker however in order to play various song. we can extend it by interfacing a keyboard for particular pattern of song.

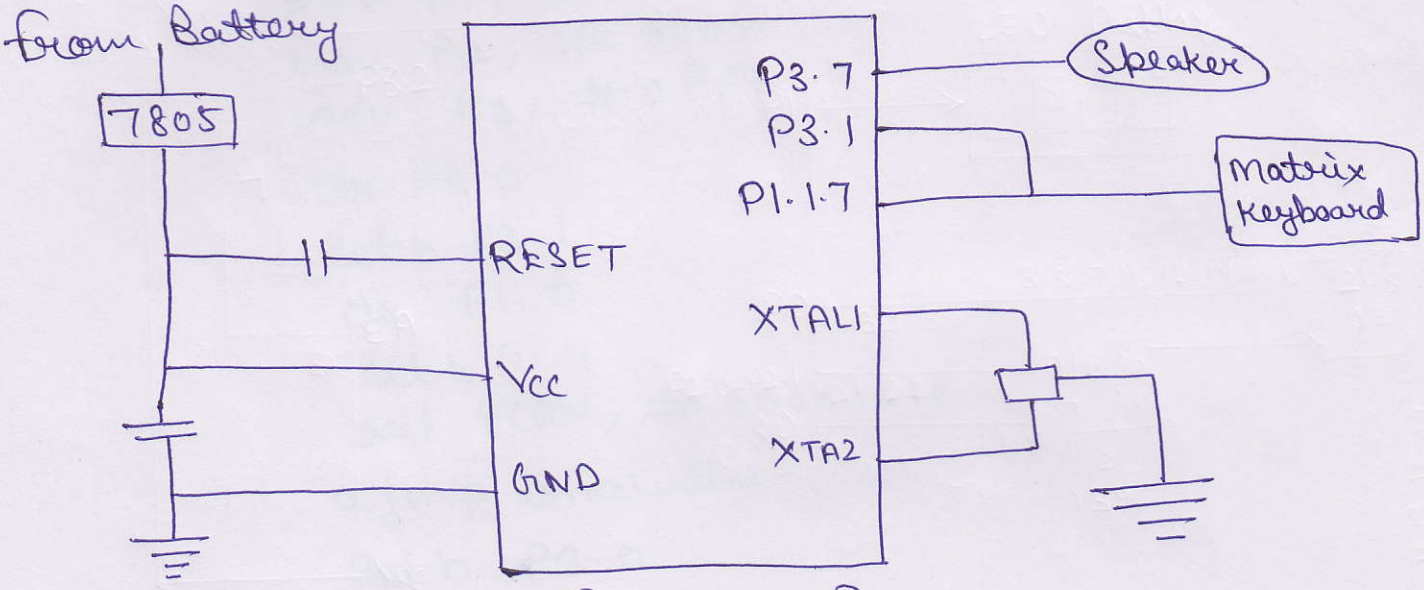


fig: Music Box

When push button is pressed, the gate of transistor would be pulled to ground, the transistor would turn OFF and reset would be high untill P1.3 becomes active high.

The timer 0 and 1 can be used to generate the music. Timer 0 is used to output the tone, while timer 1 can be used to control the timing for which music should be ON.

Timer 1 interrupt handling is to be added for switching ON and OFF the music. The keyboard is a matrix of switches which pulls down a pin of Port 1.

PROG:-

```
org 0
clr P1.7
xrl 0, #55h
xrl 1, #0AAh
```

```
xrl 3, #0FFh
mov A, R0
orl A, R1
orl A, R2
orl A, R3
jz Music
mov R0, #055h
mov R1, #0AAh
mov R2, #000h
mov R3, #0FFh
clr P3.0
Setb P3.1
clr. P1.0
Setb P1.7
orl PCON, #00000010
ajmp mainline
Setb P3.0
clr P3.1
clr P1.0
Setb P1.7
orl PCON, #%.0000010
ajmp mainline
end.
```

Mouse Wheel Turning :- The first assumption is how to run motors. If we connect same power supply which is used for microcontroller then this never works for much duration and it may burn out.

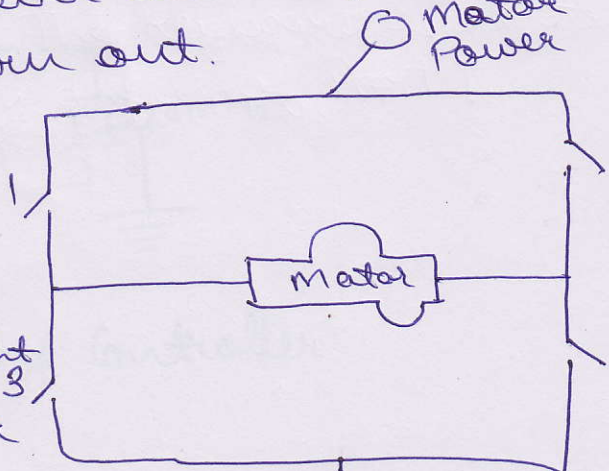
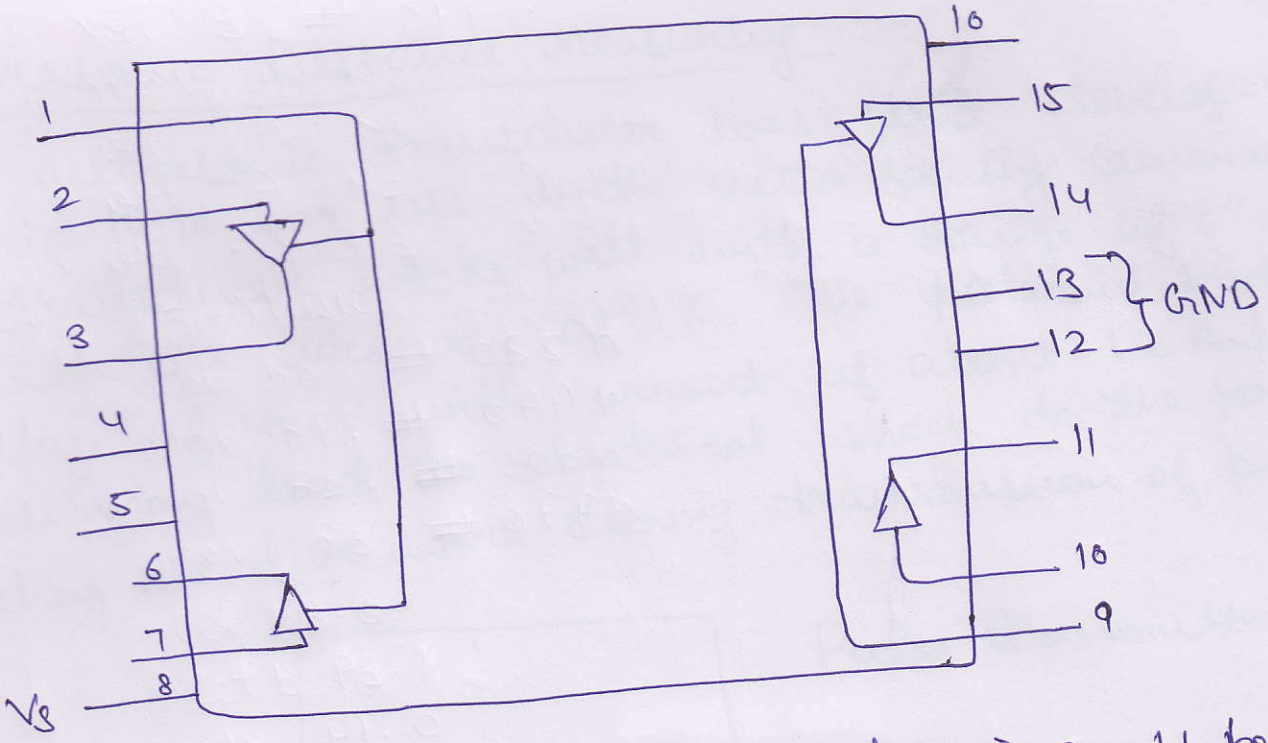
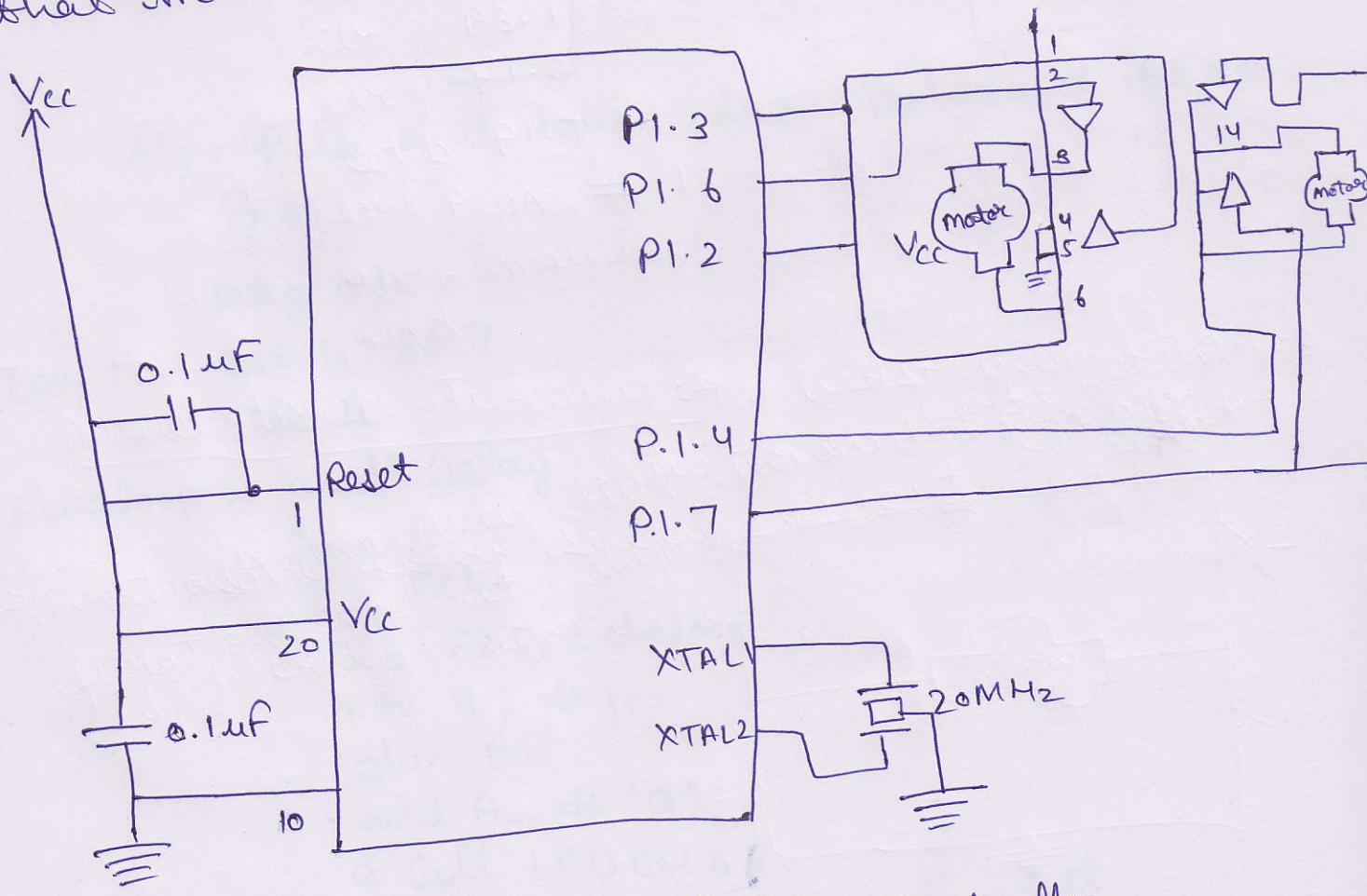


Fig: H Bridge motor driver.

If the top left and bottom right switches are closed then the motor



293D can be used to control motors in an H bridge configuration by wiring two motor power leads to each half of the chip. The 293D has a few features that make it very desirable for motor control.



Motor control using Controller.

Ultrasonic Distance Measuring :-

The ultrasonic transducer was first developed in mid 1970 for use with auto focusing cameras. This transducer works well with a range of 6" to 35" with an accuracy of 1%. The polaroid produces a voltage of 400V with current of about 1.2 Amp which may lead to electrical shock to the person touching the live wire during transmission of pulse.

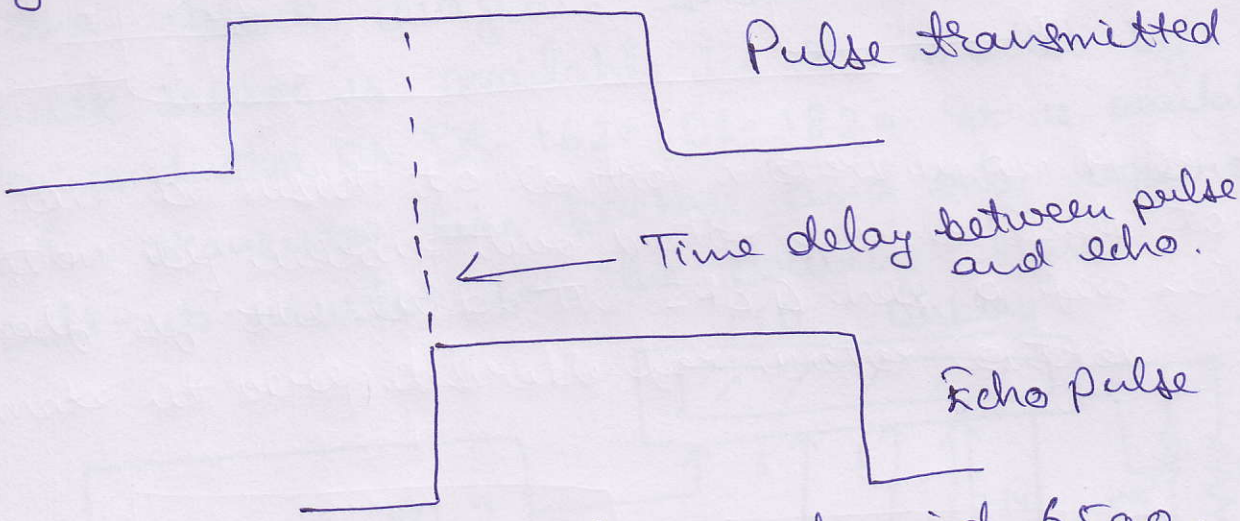


fig. Pulse and echo from polaroid 6500.

Prog :-

```
        org 0h
Loop:   Setb P3.7
        Clr A
Echoloop: acall Delay
        inc A
        Jz Echo
        Jnb P3.5, Echoloop.
        mov B, #100
        div AB
        add A, #'0'
        a Call LCDCHAR
        Clr P3.7
        ajmp Loop
Delay:  mov Count, #19
Delayloop: djnz Count, delay loop.
        ret
        end.
```

Temperature Sensor :-

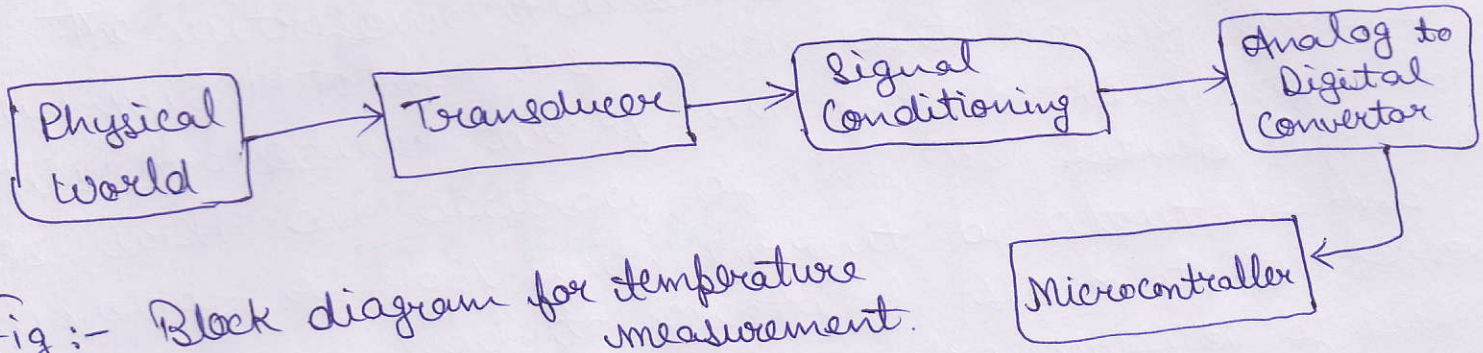


fig :- Block diagram for temperature measurement.

In the block diagram shown above the transducer sensor is available in the market by Dallas Semiconductor as DS-1620/DS-1820. It is available in a 3-Pin transistor like package and only requires a 10K pull-up resistor for the driving signal.

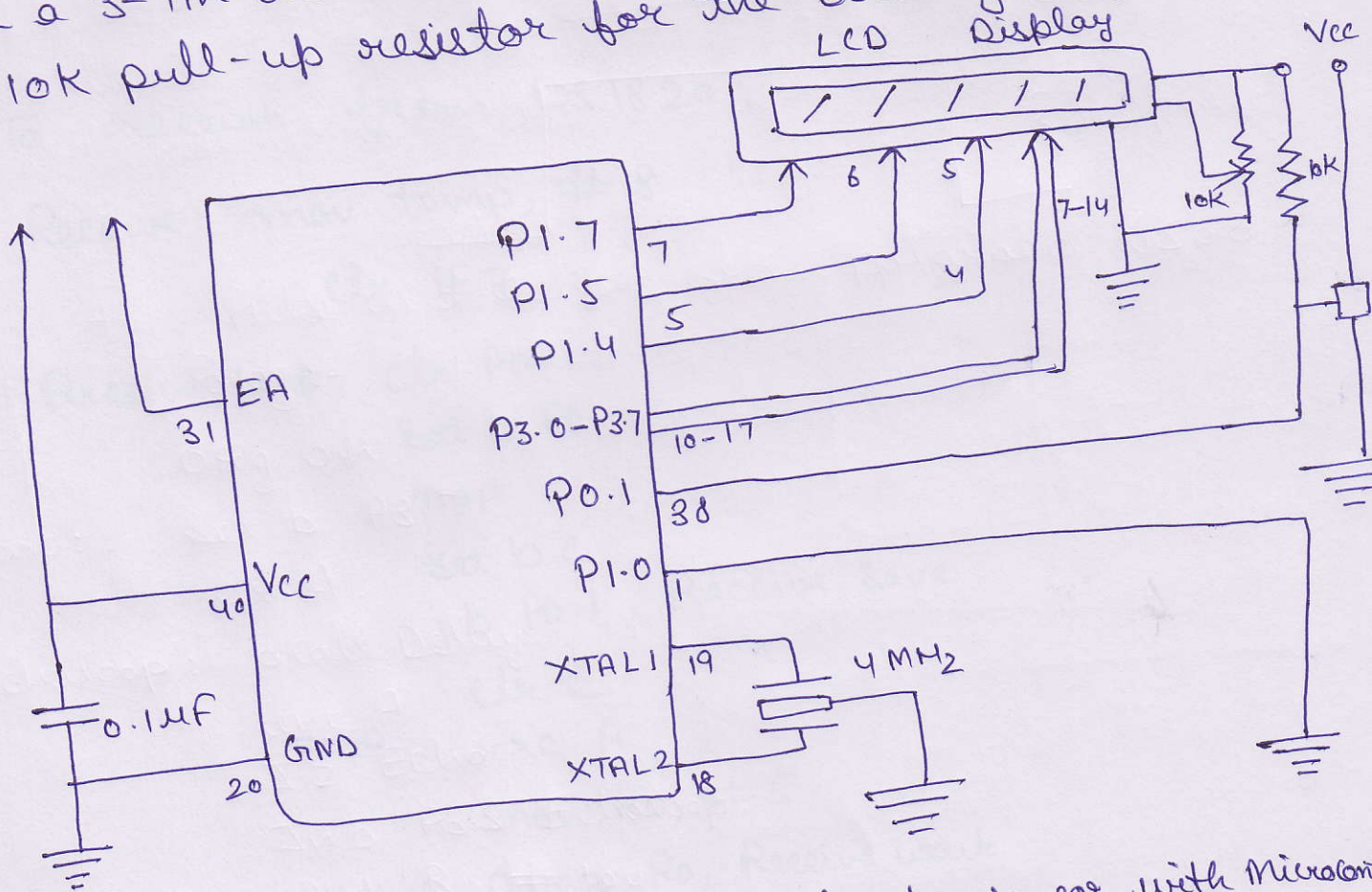


fig shows DS1820 temperature sensor with Microcon

A "0" is a low pulse of 15 μsec to 60 μsec and '1' is a low pulse greater than 1 microsecond and less than 15 microsecond.

To transmit to DS1820 :-

```
Send:       mov temp, #8
            clr IE.7
Send Loop:  rrc A
            clr P0.1
            Inc Send delay
            Set b P0.1
Send delay: mov R0, #9
Send wait:  djnz R0, Send wait
            djnz temp, Send loop.
            Set b IE.7
            ret.
```

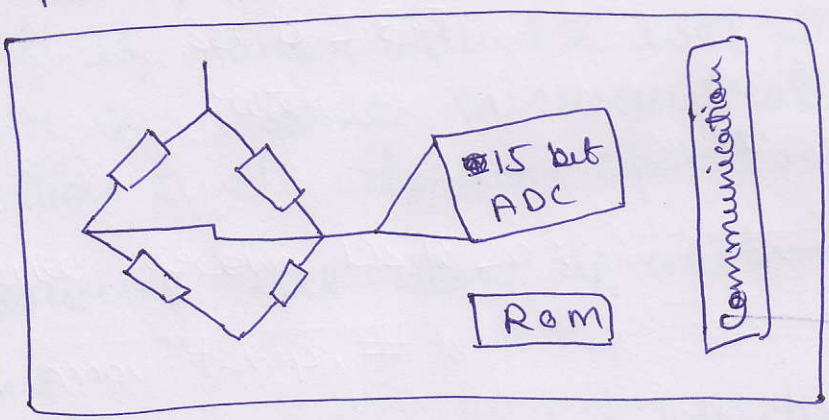
To receive from DS1820 :

```
Receive:      mov temp, #8
              clr IE.7
Receive Loop: clr P0.1
              Set b P0.1
              nop
              Set b C
              jb P0.1, Receive Save
              clr C
              rrc A
              mov R0, #5
              djnz R0, Receive wait
              djnz temp, Receive loop.
              Set b IE.7
              ret.
```

Pressure Sensor :

Following are the features of a typical Pressure Sensor:

* Integrated pressure sensor
* Pressure range 10-1100 m bar
* Small size 6.2 x 6.4 mm
* 15 Bit ADC
* 1 System clock line
* Temp. range -40 to 85°C
* Low voltage
* Low power



Above is a SMD hybrid device including a precision piezoresistive pressure sensor and an ADC interface IC. It is a miniature version of barometer/altimeter module and provides a 16-bit dataword from a pressure and temperature dependent voltage.

For the pressure measurement, the differential output voltage from the pressure sensor is converted and for temperature measurement, the sensor bridge resistor is sensed Converted.

Applications:-

* Mobile altimeter/barometer systems
* weather control systems
* GPS receivers.