```
.include “m8def.inc” #including complete details of atmega8a

.def ptsel=r10

.def glvf0=r11

.def glvf1=r12    #present glove value finger 2 (Ring finger)

.def glvf2=r13

.def glvf3=r14

.def pval0=r15     #previous glove value of finger 1 (Pinky finger)

.def pval1=r4

.def pval2=r5

.def pval3=r6

.def temp=r16      #for mathematical operations

.def temp1=r17

.def temp2=r18

.def temp3=r19

.def temp4=r20

.def temp5=r21
.def temp6=r22
.def temp7=r23
.def temp8=r24


.equ tsprt=portd    #notation for LED i.e., PORTD and pin4

.equ tled1=4


.equ brp=23    #baudrate (115200 bits per sec)

.equ refv0=$90      # for storing the reference value in the stack

.equ refv1=$91

.equ refv2=$92

.equ refv3=$93

.equ lnfd='\n’      #line feed and carriage return

.equ crrt='\r'
```

```
.equ vlw=$01      # 01 when low

.equ vnr=$00         #00 when normal

.equ vhg=$02     #02 when high

.equ gpvl=$06        #gap value (difference between the value)


.org $00             #start  (main( ))
rjmp RESET

.org $30             #left untill 30 address for usual subroutine to run


RESET: ldi temp,high(RAMEND)     #common procedures for all
microcontroller

out SPH,temp
ldi temp,low(RAMEND)

out SPL,temp
ldi temp,$ff
out ddrb,temp        #initialising port B as output or nill
ldi temp,$00
out ddrc,temp           #initialising port C as input (ADC is present there)
ldi temp,$f0
out ddrd,temp              #initialising port D as output or nill
clr temp
out portb,temp
out portc,temp
inc temp
out portd,temp




rcall urtini      #initialising the UART or baud rate to 115200 and to activate the hardware c
onnected to microcontroller

rcall adcset      #Initialising ADC before running

rcall blink1

clr ptval
rcall adcctr

rcall adcctr

rcall adcctr

sts refv0,pval0             #Storing the Inital values of flex sensor   (stored as VinX255/5
) i.e., digital value
```

```
sts refv1,pval1

sts refv2,pval2

sts refv3,pval3

rcall blink1


here:rcall adcctr           #inifinite procedure to check the change in value and to send t
he value to bluetooth
mov temp,glvf0

add temp,temp7

mov xh,temp             #Pinky

mov temp,glvf1

ldi temp7,$30           #Converting to string by adding 30 to hex value

add temp,temp7

mov xl,temp

mov temp,glvf2  #Ring

ldi temp7,$30

add temp,temp7

mov yh,temp  #Middle

mov temp,glvf3

ldi temp7,$30

add temp,temp7

mov yl,temp  #Forefinger

rcall btsend

rcall delay

rjmp here


uritini : ldi temp,high(brp)         #setting baudrate

out ubrrh,temp
ldi temp,low(brp)
out ubrrl,temp
ldi temp,(1<<RXEN)|(1<<TXEN)        #enabling bluetooth transmittor and r
eceiver
out ucsrb,temp
```

```
ldi temp,(1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0)    #enabling UART flags
(builtin)

out ucsrc,temp
rcall delay

ret
```

```
btsend: ldi temp,$35   #1st digit i.e., 5  (delimiter)

rcall srsnd

mov temp,xh
rcall srsnd

mov temp,xl
rcall srsnd
mov temp,yh
rcall srsnd
mov temp,yl
rcall srsnd
ldi temp,$36          #last digit i.e., 6
rcall srsnd
rcall sndcmd
ret
```

```
adcset: ldi temp6,$80  #company standards for adcset and adcrun (just include temp6 or
r22 register)

out adcsra,temp6

ldi temp6,$60

out admux,temp6

ret


adcrun: in temp6,admux

andi temp6,$60

or temp6,ptsel

out admux,temp6

in temp6,adcsra
```

```
ori temp6,$40

out adcsra,temp6

as01:in temp6,adcsra
sbrc temp6,adsc

rjmp as01
in temp6,adch

mov ptval,temp6

ret
```

```
adcctr: ldi temp6,$05                #Taking single value from the glove a
nd measuring it.
mov ptsel,temp6

rcall adcrun

lds temp8,refv0

mov temp7,ptval

rcall msure                  #measure the value change
mov glvf0,temp6

mov pval0,ptval

ldi temp6,$04

mov ptsel,temp6

rcall adcrun

lds temp8,refv1

mov temp7,ptval

rcall msure
mov glvf1,temp6

mov pval1,ptval

ldi temp6,$03

mov ptsel,temp6

rcall adcrun
```

```
lds temp8,refv2

mov temp7,ptval

rcall msure
mov glvf2,temp6

mov pval2,ptval

ldi temp6,$02

mov ptsel,temp6

rcall adcrun

lds temp8,refv3

mov temp7,ptval

rcall msure

mov glvf3,temp6

mov pval3,ptval

ret




msure: sub temp8,temp7

brlt mr01     #less than then send 01

cpi temp8,gpvl

brgt mr02     #greater than then send 02

ldi temp6,vnr

rjmp mrgo

mr02:ldi temp6,vhg

rjmp mrgo

mr01:ldi temp6,vlw

mrgo: ret
```

```
sndcmd: ldi temp,crrt  #sending linefeed and carriage return i.e., \n and \r for bluetooth to show end of string

rcall srsnd

ldi temp,lnfd

rcall srsnd

ret

srsnd:  nop
sd1: sbis ucsra,udre  #wait untill this register is empty before sending

rjmp sd1
out udr,temp
ret



delay : rcall dly50ms      #Standard procedure for delay given by the atmel studio

dec temp5

brne delay
ret


dly50ms: ldi temp4,$0a

d50: ldi temp1,$8f

ldi temp3,$9d

rcall dly

dec temp4

brne d50

ret

dly : nop

lp2:mov temp2,temp3

lp1:dec temp2
brne lp1
dec temp1
brne lp2
ret
```

```
blink1:ldi temp6,$08          #Procedure for blinking the led

lop14:ldi dcnt1,$02

lop13:rcall dly50ms

dec dcnt1
brne lop13              #For activating and deactivating of LED

sbic tsprt,tled1

rjmp nxt11
sbi tsprt,tled1

rjmp bnxt11
nxt11:cbi tsprt,tled1

bnxt11:dec temp6
brne lop14
cbi tsprt,tled1

ret
```