```
;*****************************************************
;Softcode For Digital Speedometer
;Made By: Abhishek Jain
;Dated: 25/02/2005
;*****************************************************

;*****************************************************
;Including Controller Definetion File
;*****************************************************

.include "2313def.inc"

;*****************************************************
;Description
;*****************************************************
;The connection details are as given in schematic
;LCD is connected to port B and sensor is connected
;to INT 0 pin of MCU

;*****************************************************
;calculation Details (Formulae Used)
;*****************************************************

;#############For speed measurement##################
;speed = 2*pi*R*3600*clk/time_count*1000*x
;R = radius of the wheel in cm (In this design R=25cm)
;time_count = value of timer counter in b/w
;two successive interrupts
;x = clock prescaler used (in this design x=1024)
;using this we have calculated the equivalent value of
;all the constants (except time_count all other parameters
;are constants)
;sh and sl are binary equivalent of this constant

.equ sh = $56
.equ sl = $49
.equ sht = $57          ;sht = sh + 1

;#############For distance measurement################
;After every 100 m the registers holding distance values
;are incremented. For this we have used following formulae
; 2*pi*R*n/100 = 100
```

```
;R = radius of the wheel in cm (In this design R=25cm)
;n is the count which signifies that 100 m has been completed

.equ n = $40

;*********************************************
;To give useful names to Registers
;*********************************************

;##########temporay registers##############
.def temp=r16
.def temp1=r18
.def c1=r22            ;used for speed calculation and for display purpose
.def c2=r4             ;used for speed calculation and for display purpose
.def d1=r5             ;used for speed calculation and for display purpose
.def d2=r6             ;used for speed calculation and for display purpose
.def status=r7  ;holds the value of status register in case of interrupt

;#########Registers used for timing########
.def count_1=r17       ;
.def time_1=r1
.def time_0=r2

;#########Registers used for speed measurement##########
.def sdigit_01=r19
.def sdigit_1=r20      ;holds the binary value of speed

;########Registers used for Distance measurement#########
.def ddigit_01=r8      ;Holds binary value of distance (0.0 to 9.9)
.def ddigit_1=r9       ;Holds binary value of distance (10.0 to 999.9)
.def ddigit_10=r10     ;Holds binary value of distance (1000.0 to 99999.9)
.def dist_count=r13

;#########Registers used for LCD display##########
.def lcd_cmd=r21
.def lcd_dat=r23
.def count=r3

;########Registers used for delays############
.def low_del=r24
.def hi_del=r25

;########Registers used for EEPROM############
```

```
.def eep_reg=r11
.def eep_addr=r12

;r14r15

;***********************************************
;Code Segment
;***********************************************

.cseg

.org 0
   rjmp    RESET

.org INT0addr
       rjmp IntV0

reti
reti
reti
reti

.org OVF0addr
       rjmp TimerV0

reti
reti
reti
reti

.org $000b

;***********************************************
;Reset interrupt subroutine
;to be executed on Reset interrupt
;***********************************************

RESET:

;#######Initializing Stack Pointer############

       ldi temp,RAMEND
   out SPL, temp
```

```
;#######Initializing I/O Ports################

   ldi temp, 0b11111111   ;configure PORT B for all outputs
   out DDRB, temp
   ldi temp,0
   out DDRD,temp
   ldi temp,$ff
   out PORTD,temp            ;configure PORT D for all inputs

;#########Initializing INT 0 interrupt########

      ldi temp,$40          ;Enabling INT 0 interrupt
      out GIMSK,temp
      ldi temp,$03          ;Interrupt on rising edge
      out MCUCR,temp

;########Intializing timing process##########
      ldi temp,$02
      out TIMSK,temp        ;Enabling timer overflow interrupt
      ldi temp,$05
      out TCCR0,temp

      ldi temp,$00          ;Initializing Registers
      out TCNT0,temp

      clr time_0
      clr time_1

;#############Initializing registers holding speed############
      clr sdigit_1
      clr sdigit_01

;#############Initializing registers holding distance############
      clr dist_count
      clr ddigit_01
      clr ddigit_1
      clr ddigit_10
      rcall eeprom_read

;############Initializing LCD Display##################
      rcall init_lcd
      rcall init_lcd
```

```
	rcall hi_delay

;############enabling global mask interrupt enable#########
	sei

;*****************************************************************
;Main Part of the Program
;*****************************************************************

main_loop:
	rcall dist		;Subroutine to calculate Distance
	rcall print_lcd ;Subroutine for LCD Display
	rjmp main_loop

;*****************************************************************
;INT 0 ISR
;*****************************************************************

IntV0:
	in status,SREG
	in time_0,TCNT0
	mov time_1,count_1
	clr count_1
	out TCNT0,count_1
	inc dist_count
	rcall speed		;Subroutine for speed calculation
	out SREG,status
	reti

;*****************************************************************
;Timer overflow ISR
;*****************************************************************

TimerV0:
	in status,SREG
	inc count_1
	cpi count_1,sht
	brsh timeff
	out SREG,status
	reti

timeff:
	clr count_1
```

```
        rcall time00
        ret


;*****************************************************
;Calculation of speed
;*****************************************************

speed:
        push d1
        push d2
        push c1
        push c2
        ldi temp1,$00
        cp time_1,temp1
        breq time0
speedq:
        clr sdigit_1
        ldi temp1,sh
        mov c1,temp1
        ldi temp1,sl
        mov c2,temp1
        rjmp speedcal

time0:
        cp time_0,temp1
        breq time00
        rjmp speedq

time00:
        clr sdigit_1
        clr sdigit_01
        pop d1
        pop d2
        pop c1
        pop c2
        ret

speedcal:
        cp c1,time_1
        brsh speed_cal
        rjmp speed0
speed_cal:
```

```
        cp c2,time_0
        brsh speed_cal1
        cp c1,time_1
        breq speed0
        rjmp speed_cal2
speed_cal1:
        inc sdigit_1
        sub c1,time_1
        sub c2,time_0
        rjmp speedcal

speed_cal2:
        inc sdigit_1
        sub c1,time_1
        sub c2,time_0
        dec c1
        rjmp speedcal

speed0:
        rjmp scal
scal:
        mov d1,c1
        mov d2,c2
        ldi temp1,10
        rjmp scal1
scal1:
        cpi temp1,1
        brne scal2
        rjmp scal3
scal2:
        dec temp1
        add d2,c2
        adc d1,c1
        rjmp scal1
scal3:
        cp d1,time_1
        brsh scal4
        rjmp speed1
scal4:
        cp d2,time_0
        brsh scal5
        cp d1,time_1
        breq speed1
```

```
	rjmp scal6
scal5:
	inc sdigit_01
	sub d1,time_1
	sub d2,time_0
	rjmp scal3
scal6:
	inc sdigit_01
	sub d1,time_1
	sub d2,time_0
	dec d1
	rjmp scal3
speed1:
	pop d1
	pop d2
	pop c1
	pop c2
	ret

;***********************************************************
;Calculation of distance
;***********************************************************

dist:
	ldi temp1,n
	cp dist_count,temp1
	brsh dist_cal
	ret
dist_cal:
	clr dist_count
	inc ddigit_01

	ldi temp,1
	mov eep_addr,temp
	mov eep_reg,ddigit_01
	rcall eeprom_write

	ldi temp1,100
	cp ddigit_01,temp1
	brsh dist_cal1
	ret
dist_cal1:
	clr ddigit_01
```

```
	inc ddigit_1

	ldi temp,1
	mov eep_addr,temp
	mov eep_reg,ddigit_01
	rcall eeprom_write
	ldi temp,2
	mov eep_addr,temp
	mov eep_reg,ddigit_1
	rcall eeprom_write

	ldi temp1,100
	cp ddigit_1,temp1
	brsh dist_cal2
	ret
dist_cal2:
	clr ddigit_1
	inc ddigit_10

	ldi temp,1			;storing data to EEPROM
	mov eep_addr,temp
	mov eep_reg,ddigit_1
	rcall eeprom_write
	ldi temp,2
	mov eep_addr,temp
	mov eep_reg,ddigit_01
	rcall eeprom_write
	ldi temp,3
	mov eep_addr,temp
	mov eep_reg,ddigit_10
	rcall eeprom_write

	ldi temp1,100
	cp ddigit_10,temp1
	brsh dist_cal3
	ret
dist_cal3:
	clr dist_count
	clr ddigit_01
	clr ddigit_1
	clr ddigit_10
	ret
```

```
;*********************************************
;Initializes distance registers values to
;previously stored value in EEPROM
;*********************************************
eeprom_read:

	cli
	push temp
	ldi temp,1
	mov eep_addr, temp
	rcall eep_nr
	mov ddigit_01, eep_reg

	ldi temp,2
	mov eep_addr, temp
	rcall eep_nr
	mov ddigit_1, eep_reg

	ldi temp,3
	mov eep_addr, temp
	rcall eep_nr
	mov ddigit_10, eep_reg

	pop temp
	sei
	ret

;*********************************************
;Subroutine to read data from eeprom
;eep_addr holds the address of eeprom
;eep_reg holds the data read
;*********************************************
eep_nr:
	sbic EECR,EERE
	rjmp eep_nr
read:
	out EEAR, eep_addr
	sbi EECR, EERE
	nop
	nop
	in eep_reg, EEDR
	ret
```

```
;***********************************************
;Subroutine to write data to eeprom
;eep_addr holds the address of eeprom
;eep_reg holds the data to be written
;***********************************************
eeprom_write:
	sbic EECR, EEWE
	rjmp eeprom_write
write:
	out EEAR,eep_addr
	out EEDR,eep_reg
	cli
	sbi EECR,EEMWE
	sbi EECR,EEWE
	sei
	ret

;************************************************************
;Init_Lcd: Initializes the 16 X 2 LCD module in 4 bit data
;transfer mode
;************************************************************
init_lcd:
   ldi lcd_cmd, 3
   rcall lcd_low_cmd
   rcall hi_delay

   ldi lcd_cmd, 3
   rcall lcd_low_cmd
   rcall low_delay

   ldi lcd_cmd, 3
   rcall lcd_low_cmd
   rcall low_delay


   ldi lcd_cmd, $28    ; set 4-bit interface
   rcall lcd_all_cmd

   ldi lcd_cmd, 8            ; set DDRAM address to 00
   rcall lcd_all_cmd

   ldi lcd_cmd, $0c
```

```
   rcall lcd_all_cmd

   ldi lcd_cmd, 6
   rcall lcd_all_cmd   ; mode setting
   ret

;*****************************************************************
;Print_Lcd: Prints speed and distance on the LCD display module
;*****************************************************************

print_lcd:

      ldi lcd_cmd, $80
   rcall lcd_all_cmd

      ldi temp, 16
   mov count, temp
   ldi ZH, high(msg1*2)
   ldi ZL, low(msg1*2)

more1:
      lpm
   mov lcd_dat, r0
   rcall lcd_all_dat
   adiw ZL, 1
   dec count
      ldi temp, 0
   cp count, temp
   brne more1

   ldi temp, 16
      mov count,temp
   ldi lcd_cmd, $c0
   rcall lcd_all_cmd
more2:
      lpm
   mov lcd_dat, r0
   rcall lcd_all_dat
   adiw ZL, 1
   dec count
      ldi temp, 0
   cp count, temp
   brne more2
```

```
;############Binary to BCD conversion######################
        clr d2
speeddigit:                     ;hundredth place value of speed
        cpi sdigit_1,100
        brsh sdigit_cal
        rjmp print1

sdigit_cal:
        inc d2
        subi sdigit_1,100
        rjmp speeddigit

print1:
        ldi lcd_cmd,$86
        rcall lcd_all_cmd
        mov lcd_dat,d2
        andi lcd_dat,$0f
        ori lcd_dat,$30
        rcall lcd_all_dat
        rcall low_delay

        mov c1,sdigit_1
        rcall bcd
        mov sdigit_1,c1
        mov c2,d1

        ldi lcd_cmd,$87         ;tenths place value of speed
        rcall lcd_all_cmd
        mov lcd_dat,c2
        andi lcd_dat,$0f
        ori lcd_dat,$30
        rcall lcd_all_dat
        rcall low_delay

        ldi lcd_cmd,$88         ;units place value of speed
        rcall lcd_all_cmd
        andi sdigit_1,$0f
        mov lcd_dat,sdigit_1
        ori lcd_dat,$30
        rcall lcd_all_dat
        rcall low_delay
```

```
    ldi lcd_cmd,$8a       ;decimal place value of speed
    rcall lcd_all_cmd
    andi sdigit_01,$0f
    mov lcd_dat,sdigit_01
    ori lcd_dat,$30
    rcall lcd_all_dat
    rcall low_delay

;##################prints distance##############################

    mov c1,ddigit_10
    rcall bcd

    ldi lcd_cmd,$c5       ;ten thousandth place of distance
    rcall lcd_all_cmd
    rcall low_delay
    mov lcd_dat,d1
    andi lcd_dat,$0f
    ori lcd_dat,$30
    rcall lcd_all_dat
    rcall low_delay


    ldi lcd_cmd,$c6       ;thousandth place of distance
    rcall lcd_all_cmd
    rcall low_delay
    mov lcd_dat,c1
    andi lcd_dat,$0f
    ori lcd_dat,$30
    rcall lcd_all_dat
    rcall low_delay

    mov c1,ddigit_1
    rcall bcd

    ldi lcd_cmd,$c7       ;hundredth place value of distance
    rcall lcd_all_cmd
    rcall low_delay
    mov lcd_dat,d1
    andi lcd_dat,$0f
    ori lcd_dat,$30
    rcall lcd_all_dat
    rcall low_delay
```

```
    ldi lcd_cmd,$c8      ;Tenth place value of distance
    rcall lcd_all_cmd
    rcall low_delay
    mov lcd_dat,c1
    andi lcd_dat,$0f
    ori lcd_dat,$30
    rcall lcd_all_dat
    rcall low_delay

    mov c1,ddigit_01
    rcall bcd

    ldi lcd_cmd,$c9      ;Units place value of distance
    rcall lcd_all_cmd
    rcall low_delay
    mov lcd_dat,d1
    andi lcd_dat,$0f
    ori lcd_dat,$30
    rcall lcd_all_dat
    rcall low_delay

    ldi lcd_cmd,$cb      ;decimal place of distance
    rcall lcd_all_cmd
    rcall low_delay
    mov lcd_dat,c1
    andi lcd_dat,$0f
    ori lcd_dat,$30
    rcall lcd_all_dat
    rcall low_delay

    ret

;************************************************************
;Binary to BCD conversion
;************************************************************
bcd:
    clr d1
bcddigit:
    cpi c1,10
    brsh bcd_cal
    ret
```

```
bcd_cal:
        inc d1
        subi c1,10
        rjmp bcddigit


;************************************************************
;Low_Delay: A 250 us delay
;************************************************************
low_delay:
        push low_del
        push hi_del
    ldi low_del, 20
ld_hi:
        ldi hi_del, 10
loop_in:
    dec hi_del
    cpi hi_del, 0
    brne loop_in
    dec low_del
    cpi low_del, 0
    brne ld_hi
    pop hi_del
    pop low_del
    ret

;************************************************************
;Hi_Delay: A 5 ms delay
;************************************************************
hi_delay:
    ldi low_del, 25
more_call:
    rcall low_delay
    dec low_del
    cpi low_del, 0
    brne more_call
    ret

;************************************************************
;Lcd_Low_Cmd: Sends a command to higher 4 bits of the LCD.
;The command nibble must be in the lower nibble of the
;variable 'lcd_cmd'
```

```
;************************************************************
lcd_low_cmd:
   mov temp, lcd_cmd
   lsl temp
   lsl temp
   lsl temp
   lsl temp
   andi temp, $f0
   out PORTB, temp
   ori temp, $08
   out PORTB, temp
   andi temp, $f7
   out PORTB, temp
   ret

;************************************************************
;Lcd_All_Cmd: sends an 8 bit command to the LCD
;************************************************************
lcd_all_cmd:
   push lcd_cmd
   lsr lcd_cmd
   lsr lcd_cmd
   lsr lcd_cmd
   lsr lcd_cmd
   rcall lcd_low_cmd
   pop lcd_cmd
   andi lcd_cmd, $0f
   rcall lcd_low_cmd
   rcall low_delay
   ret

;************************************************************
;Lcd_All_Dat: Sends a data byte to the LCD. The data byte
;is stored in variable 'lcd_dat'
;************************************************************
lcd_all_dat:
   push lcd_dat
   lsr lcd_dat
   lsr lcd_dat
   lsr lcd_dat
   lsr lcd_dat
   andi lcd_dat, $0f
   rcall lcdlowdat
```

```
   pop lcd_dat
   andi lcd_dat, $0f
   rcall lcdlowdat
   rcall low_delay
   ret

;*************************************************************
;Lcd_Low_Dat: Sends a data nibble to higher 4 bits of the
;LCD. The data nibble must be in the lower nibble of the
;variable 'lcd_dat'
;*************************************************************
lcdlowdat:
   mov temp, lcd_dat
   lsl temp
   lsl temp
   lsl temp
   lsl temp
   andi temp, $f0
   ori temp, $04
   out PORTB, temp
   ori temp, $08
   out PORTB, temp
   andi temp, $f7
   out PORTB, temp
   ret

;*************************************************************
;Message #1
;*************************************************************
msg1: .db "SPEED=   . Km/hrDIST=     . Km  "
;         12345678123456781234567812345678
```