CHAPTER 3
CHAPTER-3

ABSRTACT

Microcontrollers are becoming an integral part of engineering design process known as Embedded System, the design includes major portion of the hardware and entire software incorporated on a single chip.

The present work describes the design and development of an embedded system for automatic potentiometric titration using AVR microcontroller. Software is developed in assembly language. The software is dumped on the flash memory of microcontroller. The AVR microcontroller ATmega 8L perform data acquisition and stores the data in the on-chip RAM. After completion of data acquisition, the microcontroller sends data to PC on parallel port. The PC performs the interpretation and displays the results graphically.
3.1 Introduction

When performed manually, pH titrations are tedious and time consuming as well as prone to human error. An automatic titrator gives a solution to the time problem and moreover affords increased accuracy. Auto titrators are dedicated instruments, often incorporating integrated microprocessor/microcontroller. Auto titrators are used in a wide range of applications including process and quality control.

In automatic potentiometric titrations it is necessary to have an automated burette to add the titrant continuously in the form of aliquots. In this work a syringe driven by a stepper motor was used to deliver titrant at a constant rate [1, 2]. The stepper motor is controlled by the ATmega 8L microcontroller.

Auto titrators developed by different authors and commercially available ones are reviewed in chapter 2. From the literature review, it is found that micro controller - based automatic titrators are rare and the ones that are available commercially are PC-based, and expensive. Keeping these aspects in the view point, a mixed signal controller based automatic titration system is developed in the present work. Salient features of the proposed system are:

* Simple but yet effective design
* Low-cost and accurate
* Easy trouble-shooting
This chapter describes the hardware, software details, and the calibration of the step burette developed in this work.

3.2 Hardware description

The block diagram of the entire assembly is shown in figure 3.1. The hardware developed in the present work is divided into three parts:

Part-A: **pH titration assembly with data acquisition electronics**

Part-B: **Mechanical arrangement and associated electronic circuit for piston movement**

Part-C: **Microcontroller assembly with printer port connector to PC**

Description of the individual parts is given below:

**Part-A: pH titration assembly with data acquisition electronics**

Continuous process monitoring and control of pH requires a specially - prepared electrode designed to allow hydrogen ions (H\(^+\)) in the solution to migrate through a selective barrier, producing a measurable potential difference (voltage) proportional to the solution’s pH. Usually 59.2mV is generated per unit pH change at 25\(^\circ\)C [3]. A filling solution picks up the signal from the special pH glass electrode. A pure silver wire dipped in silver chloride passes the signal from the solution whose pH is being measured to the cable connector (BNC). This is depicted in figure 3.2. The other electrode, which is usually called as reference electrode is made from a chemical solution of neutral pH buffer solution (pH 7), which is allowed to exchange ions with the process solution.
FIGURE 3.1 Block Diagram of the Embedded Potentiometric titration system

Chapter -3 Hardware and Software details

---

Pb Electrode

Data Acquisition Electronics

Syringe with Mechanical Drive Assembly

Stepper Motor Electronics

Output Port

Input Port PCO

Microcontroller ATMega16

Printer Port

---

70
FIGURE 3.2 Internal Parts of the pH Electrode

- Wire connection point
- Glass body
- Measurement Electrode
- Silver wire
- Silver chloride tip
- Very thin glass bulb, chemically “doped” with lithium ions so as to react with hydrogen ions outside the bulb.
- Voltage produced across thickness of glass membrane
- Bulb filled with potassium chloride "buffer" solution
through a porous separator. A specially designed combination electrode and a reference electrode built into a single housing is used in the present work (figure 3.3). Its size is very small [4].

The hardware details of the embedded system designed in the present work is shown in figure 3.4. This circuit diagram consists of five major blocks. These are:

- Amplifier section associated with pH electrode system
- Microcontroller section
- Stepper motor driver interface
- Printer interface
- Battery back-up unit

The output of the pH electrode is connected to a BNC connector using a minimal length shielded cable. It is connected to the non-inverting input of Op-Amp1 (CA3140), configured as a voltage follower. The pH and reference inputs go into the Op-Amp1 due to the very high resistance of the pH glass. The output of Op-Amp1 is connected to Op-Amp2 (TL081) configured as a non-inverting amplifier. The gain of the amplifier is given by

$$\text{Gain} = 1 + \frac{R_F}{R_1} = 1 + \frac{10\text{K}\Omega}{1\text{K}\Omega} = 11$$

The capacitor $0.1\mu\text{F}$ is connected across the feedback resistor of Op-Amp2 in order to minimize the oscillations. The output of Op-Amp2 may be positive or negative depending on the pH value of the solution.

Scale changing of the voltage from the output of Op-Amp2 is accomplished by using a special voltage divider network, so that the input to the on-chip ADC will always be positive, irrespective of the polarity at the output of Op-Amp2.
Figure 3.3 specially designed pH electrode
Chapter -3

Hardware and Software details

FIGURE 3.4: Embedded Circuit of the System
Part-B: **Mechanical Arrangement and Associated Electronic Circuit for Piston Movement**

The mechanical arrangement plays an important role in the delivery of aliquots of titrant to the analyte solution taken in a beaker. A specially designed plastic mould is taken and a hypodermic syringe is mounted on it. The piston of the syringe is driven by a rack-and-pinion arrangement driven by a stepper motor. The stepper motor rotational motion is converted to a linear translational motion, which pushes the piston of the syringe at a constant rate. Thus the titrant is added to the analyte continuously at a constant rate. The entire setup is mounted on a wooden plank as shown in figure 3.5.

Stepper motors are used in different types of applications. These applications include paper driving mechanism in printers, X-Y recorders, Floppy drive read/write head movement mechanism, Platten movement in Hard disk, Robotic arms etc. Small stepper motors with simple driver hardware are indeed desirable in embedded systems. In the present work a stepper motor removed from discarded 5 ¼ inches floppy drive mechanism is used. Such stepper motors are also available commercially as shown in figure 3.6. This is a tiny stepper motor and gives a stepping angle of 1.8 degree in full-step and 0.9 degree in half-step mode. Further, mounting and handling of the motor are easy. The average maximum current drawn by the motor is approximately 200mA. As the output port of the microcontroller cannot drive the stepper motor directly, Darlington array in the form of IC (ULN 2003A) is used to drive the stepper motor. The pin configuration of ULN 2003A is shown in figure 3.7. ULN 2003A is high voltage, high current Darlington arrays each containing seven open collector Darlington pairs with
common emitters. Each channel is rated at 500mA and can withstand peak currents of 600mA. Suppression diodes are included for inductive load driving and the inputs are pinned opposite the outputs to simplify board layout [5, 6, and 7].

A magnetic stirrer is used to stir the solution in order to obtain uniform concentration. The stirring is done at low speed so that there will not be considerable change in temperature of the solution. The entire assembly is placed in a chamber which is maintained at 298K.
(a). Four phase coil setup [12,13]

(b). Stepper motor

Figure 3.6. Photograph of the Stepper motor
Figure 3.7 Pin configuration of ULN2003A
Part-C: Microcontroller Assembly with Printer Port Connector to PC

The embedded system developed in the present work uses Atmel’s AVR ATmega 8L microcontroller for data acquisition and then to send the data to PC over the printer port data lines. Reasons for choosing the ATmega 8L is as follows [8]:

* Advanced RISC architecture.
* 8K-bytes of in-system self-programmable flash
* 512-byte EEPROM
* 1K-byte internal SRAM
* 6-channel on-chip ADC with 10-bit accuracy
* External and internal interrupt sources
* 23 Programmable I/O lines [PORT-B (8), PORT-C (7), PORT-D (8)]
* Internal RC oscillator (Programmable)

The circuit diagram of the microcontroller connected to the stepper motor driver and the printer port of the PC is shown in figure 3.4. The oscillator frequency is selected by means of software which is discussed in the software section. The external hardware interrupt INTO is selected for transmitting the data stored in SRAM (on-chip) to the PC. The data is stored in the form of a file on the PC’s hard disk for further analysis.

As mentioned in Part-B, the step burette is driven by a unipolar stepper motor. The stepper motor is driven by a power amplifier IC ULN 2003. The four driving lines are
drawn from PORT-D (PD7-PD4) of the microcontroller. The four output lines coming from ULN2003 are connected to the four poles of the stepper motor.

The analog input of the on-chip, 10-bit ADC is fed from the output of Op-Amp2. The 10-bit converted data is output on two ports, namely, PORT-B (PB0-PB7) and PORT-C (PC1, PC2). These lines are connected to the data lines (D0----D7) and status lines (S3, S4) of the printer port [9], respectively. The configuration of the printer port is shown in figure 3.8. The handshake between ATmega 8L and PC is accomplished using the status line S7 of the printer port. Software is developed in C for data transfer.

In order to continue the data acquisition under the power failure condition or to retain the data in the on-chip static RAM, the microcontroller is battery backed up using the circuit shown in figure 3.9.

Software is developed in assembly language using AVR studio [10]. For cross-platform development, AVR studio and AVRGCC together gives the ability to do advanced application development. Using the studio, one can create, compile, assemble, debug and link programs written in C, C++ and AVR assembly language. The AVR studio can be obtained from www.atmel.com. It is an Integrated Development Environment (IDE) for AVR microcontrollers. It provides a complete development suite consisting of Editor, Compiler, Assembler, Debugger and Simulator. The step-by-step procedure for program development in assembly or C language is available in the help menu of the Studio.
Chapter 3

Hardware and Software details

FIGURE 3.8 Configuration Of the Printer Port

Data port
(outputs)

Port address

Pin #
2 — D9
3 — D8
4 — D7
5 — D6
6 — D5
7 — D4
8 — D3
9 — D2

Outputs

Control port
(outputs)

Port address

Pin #
1 — Strobe (Bit 0)
14 — Autofeed (Bit 1)
16 — Init prn (Bit 2)
17 — Select (Bit 3)

Inputs

Status port
(inputs)

Port address

Pin #
15 — Error (Bit 3)
13 — Selct (Bit 4)
12 — PE (Bit 5)
10 — Ack (Bit 6)
11 — Busy (Bit 7)

Other pins
18-25—GND

10 11

DB 25 Connector

FIGURE 3.8 Configuration Of the Printer Port
The software developed in the present study is downloaded onto the on-chip flash memory using a special programmer. The programmer is connected to the PC via printer port. The SPI-flash Programmer software downloads the hex file onto the chip using the printer cable. Figure 3.10 shows the special photograph of the programmer connected to the PC.

3.3 Software Details

Broadly, the software details comprises of two parts. In the first part, the data acquisition software comprises of acquiring the voltage from the electrode system and converting it to 10-bit digital form and storing it on the system's SRAM. Simultaneously, the stepper motor is driven and the magnetic stirrer is activated. This process is repeated 200 times, so that 200 sampled data are stored in SRAM. The second part of the software is an "Interrupt-driven Service Routine" (ISR). This routine sends the data stored in SRAM to the PC over the printer port. Also, included in this software is the hand shake signals between the microcontroller and PC, for correct data transmission. The software on PC is developed in C language. Thus this program together with the ISR plays an important role in accomplishing the successful data transmission between the microcontroller and PC. The 10-bit digital data is converted to its equivalent analog voltage using look-up table. This process is necessary for the graphical representation of the results.

The flowchart for the first part of the software is shown in figure 3.11. It also includes the ISR program. The flowchart for the second part of the software is shown in
Figure 3.10 Special AVR Programmer connected to the PC [14]
Figure 3.11 Flow chart for data acquisition

START

Initialize the Stack & ports, Port-D0=O/P, Port-D1=I/P, PC0=I/P, PC1 & PC2=O/P

Initialize the SRAM & ADC Channel [ADMUX, ADCSRA]

Load Stepper motor sequence for each step=1.8° and set motor in power down mode

Output the load sequence, and delay

Set SOC & delay

Transfer the 10-bitt data to the SRAM

Is the step=200?

Increment SRAM Address by one

Is location=200?

END
Figure 3.12 Flow chart for ISR

START

Initialize the DATA, STATUS and CONTROL Ports of the

Initialize an OUTPUTFILE for writing data, i = 0

Check Whether $S_7^{bit} = 1$?

Transfer 10-bit data to the file from the SRAM

Delay for 60ms

i = i+1

Is i = 200?

END
figure 3.12. The program listing of the above tasks are given in Tables 3.1 and 3.2 respectively.

3.4 Calibration of Step burette [11]

The overall accuracy and precision of the volume of titrant delivered by the automated burette are important parameters. The aliquots of water delivered by the burette of 30 ml capacity for different volumes of titrants is shown in Table 3.3. The average volume of the aliquot dispensed for each full-step and half-step are computed and is used in the computation of determining the point of equivalence.

The accuracy of the burette was determined gravimetrically with water, and found to be accurate to about 0.1% over the entire volume range.

A new setup to study the potentiometric titrations is thus developed in the laboratory using inexpensive components with considerably good accuracy. Further the calibration of the burette has clearly indicated, accuracy is obtained with the present instrument is on par with the commercially available instruments.
TABLE-3.1 Program for data acquisition

; TITRATION EXPERIMENT
.DEVICE ATmega8
.cseg
.org $000
rjmp reset
.org $001
rjmp EXT_INT0
.reset:
    cli ; Disable all interrupts
    ldi r16,$5f
    out $3d,r16
    ldi r17,$04
    out $3e,r17 ; Stack initialization
    ldi r16,$f0
    out $11,r16 ; port -D upper—>o/p, lower—>i/p
    ldi r16,$00
    out $14,r16 ; pco=ADC0IN,
    ldi r16,$00
    out $12,r16 ; stepper motor in power down mode(port_D)
    ldi r16,$40
    out $07,r16 ; ADMUX—>select CH0, Aref=AVcc
    ldi r16,$83
    out $06,r16 ; ADCSRA—>select freq. factor, enable ADC
    clr r29
    ldi r28,$60 ; start of data RAM(0060H)
    ldi r18,$32 ; 50 samples counter (50x4x2=400)
out_step:
  ldi r31, high(step*2)
  ldi r30, low(step*2)
  ldi r20, $04
  sbi $06, 6
  ldi rl6, $32
  dec rl6
  bme BACK

inn_step:
  lds r2, $0024
  lds r3, $0025
  st y, r2
  inc r28
  st y, r3
  inc r28
  cpi r28, $00
  bme proc
  inc r29

BACK:
  dec r16
  ; conversion time
  brne BACK
  lds r2, $0024
  ; load ADCL to r2
  lds r3, $0025
  ; load ADCH to r3
  st y, r2
  ; store ADCL to data memory
  inc r28
  ; store ADCH to data memory+1
  inc r28
  cpi r28, $00
  brne proc
  inc r29

proc:
  lpm
  out $12, r0
  adiw r30, 0x01
  rcall dly
  rcall dly
  rcall dly
  dec r20
  brne inn_step
  dec r18
  brne out_step
  ldi r16, $00
  ; stepper motor in power mode
  out $12, r16
  sei
  ldi r16, $c3
  ; program GCIR
Chapter -3  Hardware and Software details

out $3b,r16 ;ADDR of GICR=$3b
ldi r16,$00
out $35,r16 ;program MCUCR,tigger by low-level

here:  rjmp here
step:
  .db $60,$a0,$90,$50

dly:     ldi r23,$ff
dly_1:   ldi r24,$ff
dly_2:   dec r24
  bne dly_2
  dec r23
  bne dly_1
  ret

EXT_INT0:  cli
  ldi r16,$5f
  out $3d,r16
  ldi r17,$04
  out $3e,r17
  ldi r16,$ff
  out $11,r16 ;port-D o/p
  ldi r16,$01
  out $12,r16 ;bit D0-->HI
  ldi r16,$ff
  out $17,r16 ;port-B o/p
  ldi r16,$06
  out $14,r16 ;port-C -->o/p
  clr r29
ldi r28, $60
ldi r17, $c8 ; total RAM = 200(c8) -> 200x2 = 400

BACK_1:
    ld r2, y
    out $18, r2 ; lower byte to RAM -> port-B
    inc r28
    ld r3, y
    lsl r3 ; c2, c1
    out $15, r3 ; upper two bits -> (c2, c1)

ldi r16, $00
out $12, r16 ; bit D0 -> lo(10 bit data from microcontroller to PC

rcall dly
ldi r16, $01
out $12, r16 ; bit D0 -> HI
inc r28
cpi r28, $00
brne proc_1
inc r29

proc_1:
dec r17
brne BACK_1

stop: rjmp stop
TABLE -3.2 Program for ISR

```c
#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <conio.h>

#define D_PORT 0x378
#define S_PORT 0x379
#define C_PORT 0x37A

void main (void)
{
    unsigned int c,dl,dh;
    int i,j,k;
    FILE *fpout;
    outportb(C_PORT,0x20);
    fpout=fopen ("outputfile","w");
    for (i=0; i<200;i++)
    {
        TOP: if ((c=inportb(S_PORT)&0x80)==0x80)
        {
            dl=inportb(D_PORT);
            fputc(dl,fpout);
            dh=inportb(S_PORT)>>3&0x03;
            fputc(dh,fpout);
            delay(60);
        }
        else
        {
            goto TOP;
        }
    }
}```
 fclose(fpout);
 getchar();
 }

<table>
<thead>
<tr>
<th>Nominal Volume (ml)</th>
<th>Number of Steps</th>
<th>Delivered volume (ml)</th>
<th>Mean of delivered aliquot (μL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5</td>
<td>50</td>
<td>3.49</td>
<td>69.8</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>7</td>
<td>70</td>
</tr>
<tr>
<td>10.5</td>
<td>150</td>
<td>10.48</td>
<td>69.8</td>
</tr>
<tr>
<td>14</td>
<td>200</td>
<td>13.98</td>
<td>69.9</td>
</tr>
<tr>
<td>17.5</td>
<td>250</td>
<td>17.5</td>
<td>70</td>
</tr>
<tr>
<td>21</td>
<td>300</td>
<td>20.96</td>
<td>69.86</td>
</tr>
<tr>
<td>24.5</td>
<td>350</td>
<td>24.48</td>
<td>69.94</td>
</tr>
<tr>
<td>28</td>
<td>400</td>
<td>27.98</td>
<td>69.98</td>
</tr>
</tbody>
</table>

Mean of delivered aliquot = 69.91 = 70 μL

Table 3.3 Aliquot delivery data obtained with a 30 ml burette
References


2. CMA (centre for microcomputer applications) “Step burette”, www.CMA.science.uva.nl


4. pH Products company, Sanath nagar, Hyderabad.


10. AVR studio and AVRGCC”www.atmel.com” or www.avrfreaks.net.


14. Special Programmer from NSK Electronics, No- 66/12, S.P. Road, Bangalore.