4.2 Interfacing and Programming of ADC in Single Conversion Mode

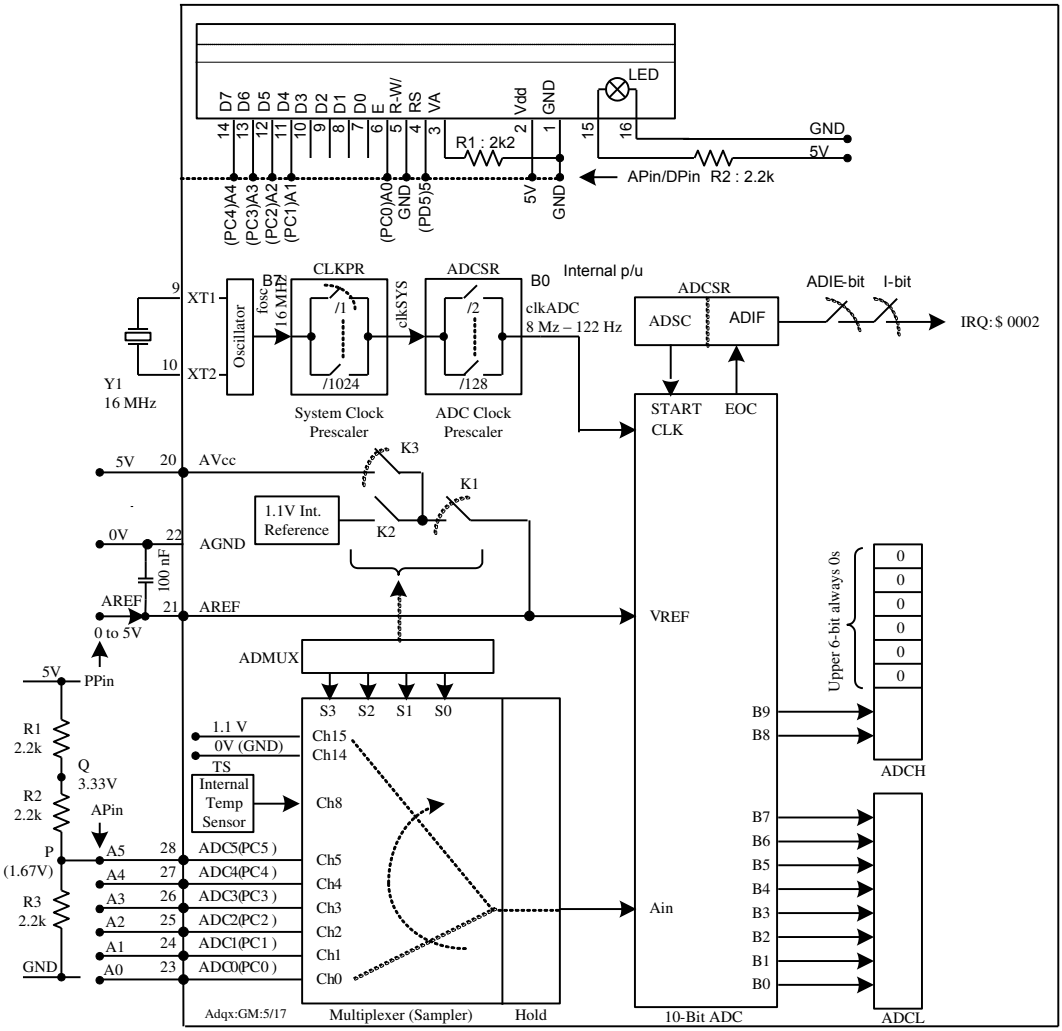

Figure-4.2: Interfacing and programming of ADC module

In this section, we will develop setup and program codes for the functional check of the ADC module of the ATmega328 Microcontroller. The Test Bench is the 'Arduino UNO Learning Kit and its associated IDE Interface.' A functional check is usually carried out in the following way: a known voltage (the excitation) is injected at the input; proportional known output (the response) is observed. In the present case, we will apply 1.67V, 3.33V, and 5.00V from the voltage divider into Ch5; proportional known output 0156h, 02A9h, and 03FFh are expected to appear on the LCD. During this operational check, we will be familiar with the procedures of putting the ADC into operation. The EOC would be sensed by polling the ADIF-bit of the ADCSRA-register.

(1) Refer to Fig-4.2, let us build a voltage divider (R1-R2-R3) circuit on the breadboard; connect the P-point voltage (1.67V) with A5-pin (Ch5 of the ADC) of the Arduino Kit.

(2) Place a LCD display unit (2-Line 16-Character) on the breadboard as per Fig-4.2, and connect it with the ATmega328 MCU using the APin/DPin connectors of the Arduino.

(3) Let us carry out initialization tasks as needed before putting the ADC into 'Single Conversion Mode.' As the input signal is a constant value, we may sample it in every 3-sec interval by calling the *delay (3000)* function of the Arduino IDE.

In single conversion mode, the ADC is started for once; wait until the conversion is complete by polling the ADIF-bit; read the ADCL and ADCH data. The ADC is not automatically started for the next time. The procedures for single conversion mode are:

1. LL → PRADC bit of PRR-register : ADC is connected with 5V
2. LH → ADEN bit of ADCSRA : ADC is enabled
   LL → ADCSC bit of ADCSRA : ADC is not started
   LL → ADATE bit of ADCSRA : Auto Triggering OFF
   LH → ADIF bit of ADCSRA : ADC Interrupt Flag cleared
   LL → ADIE bit of ADCSRA : ADC Interrupt is disabled
   [1, 1, 1] → [ADPS2:ADPS0] of ADCSRA : 16 MHz/128 = 125 KHz
3. [0, 1] → [REFS1, REFS0] of ADMUX : V_{REF} of ADC is AVcc (+5V)
   0 → ADLAR bit of ADMUX : ADC result is right adjusted
   [0, 1, 0, 1] → [MUX3:MUX0] : ADC Channel-5 is selected
4. LH → ADSC bit of ADCSRA : Start the ADC
5. Wait here until conversion is complete by polling ADSC-bit or ADIF-bit.
6. Clear ADIF (LH → ADIF bit) of ADCSRA if ADIF-bit was being polled in Step-5.
7. Read ADCL : Read Lower 8-bit first
   Read ADCH : Read upper 2-bit along with Bit-11 to Bit-15 which are 0s.

(4) Assembly Codes for the instructions of Step-3 (tested using RMCKIT/AVR Studio 4)

```
        .org    0x0040
START:  nop
L1:     ldi     r16, 0x00
        out     PRR, r16        ; 5V supply is connected to all IO modules including ADC
L2:     ldi     r16,  0x93      ; 10010011  clkADC = 138.24 KHz
        out     ADCSRA, r16
L3:     ldi     r16, 0x45       ; 0100 0101       Ch5
        out     ADMUX, r16
L4:     in      r16, ADCSRA
        ori     r16, 0x40       ; 0100 0000  ADC is started
        out     ADCSRA, r16
L5:     in      r16, ADCSRA     ; polling the ADIF bit for LH to sense End-of-Conversion
        rol     r16
        rol     r16
        rol     r16
        rol     r16
        brcc    L5
L6:     in      r16, ADCSRA
        out     ADCSRA, r16     ; ADIF flag is cleared by writing LH at ADIF-bit
L7:     in      r16, ADCL
        in      r17, ADCH       ; ADC value in <r17 r16>.
```

P425 **(5)** Arduino IDE Codes for the instructions of Step-3/4 (tested using Arduino)

```
#include <LiquidCrystal.h>
//LiquidCrystal lcd(RS, E, D4, D5, D6, D7);
LiquidCrystal lcd(5, A0, A1, A2, A3, A4);

void  setup()
{
  lcd.begin(16, 2);
  lcd.setCursor(0,0);                 //DP0 position of Top Line (L0)
  //--------------------
  pinMode(13, OUTPUT);        // Labels correspond to Step-3
  bitClear(PRR, 0);                   //L1:
  ADCSRA = 0x97;                      //L2: clkADC = 125 KHz
  ADMUX = 0x45;                       //L3: Ch5

}

void loop()
{
  lcd.clear();                                  // remove from LCD whatever is there
  bitWrite(ADCSRA, 6, HIGH);                    //L4: ADC is started by ADSC
  while (bitRead(ADCSRA, 4) != HIGH)            //L5: Checking EOC by sensing ADIF
    ;
  bitWrite(ADCSRA, 4, HIGH);                    //L6: ADIF-bit is cleared by putting LH

  int x1 = ADCL;                                //L7:
  int x2 = ADCH;                                //L7A:
  x2 = x2<<8;                                   // L7B: shifting ADCH value to the left by 8-bit
  x2 = x2 | x1;                                 // L7C: making 16-bit chunk out of ADCH and ADCL

  digitalWrite(13, !(digitalRead(13)));         //Toggling L as an indication that conversion is being taken place
  lcd.print(x2, 16);                            //ADC value in LCD in Hex format
  delay(3000);                                  //Repeat acquisition at 3-sec interval
}
```

P246 **(6)** Arduino IDE Codes for the instructions of Step-3/4/5 (tested using Arduino)

```
#include <LiquidCrystal.h>
//LiquidCrystal lcd(RS, E, D4, D5, D6, D7);
LiquidCrystal lcd(5, A0, A1, A2, A3, A4);       //takes care of user connection between LCD and Arduino

void setup()
{
    analogReference(DEFAULT);                   //does it cover tasks of L1 - L3 of Step-3?
}

Void      loop()
{
    unsigned int x = analogRead(A5);            // does it cover tasks of L4 – L7C of Step-5?
    //---------------------------
    digitalWrite(13, !digitalReda(13));         //Toggle L to see that conversion takes place
    lcd.print(x, 16);                           //ADC value in LCD in hex format
    delay (3000);                               //wait for 3-sec and then acquire input signal again
}
```

(7) **(a)** Connect P-point (1.67V) of the voltage divider of Fig-4.2 with Ch5 via APin-A5.
(b) Compile and upload P246 program of Step-6.
(c) Check that the LCD shows **a** value: close to: 155h ± 2%
(d) Connect Q-point (3.33V) with Ch5 and check that LCD shows close to 2A7h ± 2%.
(e) Connect 5V-point (500V) with Ch5 and check that LCD shows close to 3FFhh ± 2%.

P248 **(8)** Add the following two instructions with P246 just before the *delay()* function.

```
lcd.setCursor (0, 1);
lcd.print(ADCSRA, 16);
```

Save the new program as P248. Compile and upload the program. Check that the bottom line of the LCD shows: 97h. Consult data sheet of ATmega328 for the ADCSRA-register; decode the number 97h (1001 0111) and find that the clkADC has been set at 125 KHz.