4 Analog to Digital Converter (ADC)

4.1 Analog-to-Digital Converter Module of ATmega328P MCU

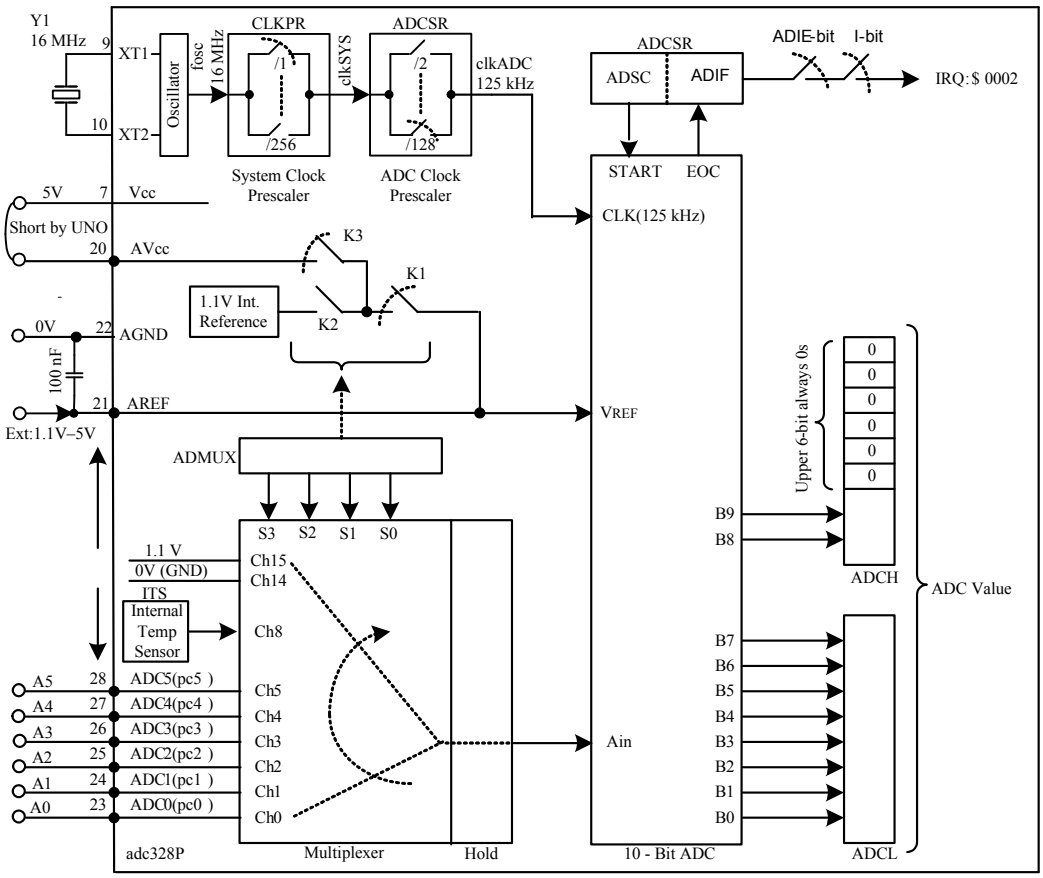

Figure-4.1: *ADC Module of ATmega328P Microcontroller*

(1) An ADC circuit converts 'analog signal' into 'digital signal'. The ADC Module of ATmega328P MCU will convert a sample of a DC voltage into 10-bit digital signal (B0 – B9).

(2) The maximum DC voltage that can be placed at the A_{in}-pin of the ADC for conversion is equal to the voltage that is placed at the V_{REF}-pin of the ADC. This voltage of the V_{REF}-pin is known as **Full Scale.** At present, the V_{REF}-pin is connected with AVcc-pin which is shorted to Vcc-pin (usually 5 V). Therefore, the range of analog signal that the ADC can convert is: 0 V to 5 V.

(3) If we apply 0 V to the ADC for conversion, the ADC will produce all 0s for the 'output 10 digital bits'; that is: B9 B8 B7 B6 B5 B4 B3 B2 B1 B0 = 0 0 0 0 0 0 0 0 0 0 = 0x000 = decimal 0.

(4) If we apply 5 V to the ADC for conversion, the ADC will produce all 1s for the 'output 10 digital bits'; that is: B9 B8 B7 B6 B5 B4 B3 B2 B1 B0 = 1 1 1 1 1 1 1 1 1 1 = 0x3FF = decimal 1023.

(5) If we apply 250 mV (0.25 V) to the ADC for conversion, the ADC will produce 0 0 0011 0011 ((1023/5)*0.25 = 51) for the 'output 10 digital bits'; that is: B9 B8 B7 B6 B5 B4 B3 B2 B1 B0 = 0 0 0 0 1 1 0 0 1 1 = 0x033 = decimal 51.

(6) If we apply an unknown voltage V_{DT} to the ADC for conversion, the ADC will produce decimal (1023/5)*V_{DT}; the digital form of this value can be known if V_{DT} is known. We will designate the output 10-bit value of the ADC by the symbolic name ADCV.

(7) In computer literature, data size goes by 8-bit (byte), 16-bit (int), 32-bit (long), and 64-bit (long long). The ADC of Fig-4.1 produces 10-bit; so, we will make it 16-bit by placing six 0s to its left.

(8) The lower 8-bit of ADCV is stored into ADCL Register; the upper 2-bit of ADCV is stored in ADCH Register; where, the upper 6-bit of ADCH Register are always 0s.

(9) ADC Module of Fig-4.1 has 9 active analog channels named Ch0 – Ch5, Ch8, Ch14-Ch15. The lower 6 channels (Ch0 – Ch5) can acquire signals from external devices via A0 – A5 pins. Here, we observe that the analog channels (Ch0 – Ch5) are the alternate functions of the port lines (pc0 – pc5) of Port-C. A port line (say: pc0) turns into analog channel (say: Ch0) when we select the channel with the help of ADMUX Register (Fig-4.1) or of *int x = analogRead(A0)* instruction.

(10) An ADC Module needs clock signal (clkADC) to convert the input analog signal. For a 10-bit ADC, there is a need of 10 clock pulses. The recommended clkADC for the ADC of the UNO Board is 50 kHz to 200 kHz. Therefore, the conversion time is from 200 µs (1/50000*10) to 50 µs. The optimum clkADC is 125 kHz, and hence the conversion time is 80 µs. If the ADC is operated beyond the limit of the given clkADC, the accuracy of the result is not guaranteed.

(11) The ADC Module of Fig-4.1 can have one of the following voltages at its V_{REF}-pin; where, the selection is made with the help of ADMUX Register or of *analogReference()* instruction.
(a) 5 V (DEFAULT) **(b)** 1.1 V (INTERNAL) **(c)** 1.1 V to 5 V (EXTERNAL) via AREF-pin.

(12) Resolution of an ADC has been defined as the weight for the LSBit which is equal to –
Resolution = $\frac{Full\ Scale}{2^n - 1} = \frac{5000\ mV}{2^{10} - 1} = 4.89\ mV$; where, *n* is the size of the ADC (here it is: 10)

(13) The ADC accuracy is up to ± 2 LSB. It means — with 5 V full scale, we will get at output of ADC a value of 510 or 514 instead of 512 ((1023/5)*2.5) when 2.50 V is applied to an analog channel.

(14) After selecting the V_{REF} voltage and the analog channel, we give 'ADC Start' command (HIGH at ADSC-bit) which comes from the ADSC-bit of ADCSR Register. The ADSC-bit remains at HIGH state as long as the conversion is going on. At the end-of-conversion (EOC), the ADSC-bit assumes LOW state. The end-of-conversion event is also known to the user by placing HIGH at the ADIF-bit of the ADCSR Register. The EOC event can also be known to the MCU through the generation of interrupt request signal.

4.2 Software Codes for the operation of the ADC Module

(1) To select 5 V at the V_{REF}-pin.
```
analogReference(DEFAULT);
```

(2) To read 10-bit of ADC value in ONE GO.
```
int x = analogRead(A0);  //upper 6-bit is always 0
```

(3) To select an analog channel (say: Ch0 via external A0-pin), start ADC, wait until the conversion is complete, read the ADC value, and then discard. This command is only valid for Ch0 – Ch5.
```
(void)analogRead(A0);
```

(4) To read lower 8-bit of ADC value from ADCL Register. This command must be executed after executing these register level commands: Start ADC, wait until conversion is complete.
```
byte x1 = ADCL;     //this code must be executed first before executing the command of Step-4.
```

(5) To read upper next 2-bit of ADC value. ADCH will return 8-bit; where, the upper 6-bit are 0s.
```
byte x2 = ADCH;
```

(6) To form 16-bit from x1 and x2.
```
int x = (int)(x2 <<8)|(int)x1;  //x2 is shifted to left by 8-bit, and then it is ORed with x1
```

(7) To read 16-bit value of the ADC at a time, the following command could be executed after executing these register level commands: Start ADC, wait until conversion is complete.
```
int x = ADCW;
```

4.3 Application of ADC

(1) Introduction: Let us practice an application of ADC Module by building Thermometer using LM35 temperature sensor and UNO. Fig-4.2 depicts the block diagram of the proposed Thermometer. The signal (V_T: DC voltage proportional to room temperature) of the LM35 sensor enters into ADC Module of the MCU. After digitization, the 10-bit data enters into MCU for processing to reconstruct the decimal value of temperature. Finally, the room temperature is shown on SM via UART Port, on LCD via I2C Bus, and on CC7SD Unit via DIO Port.

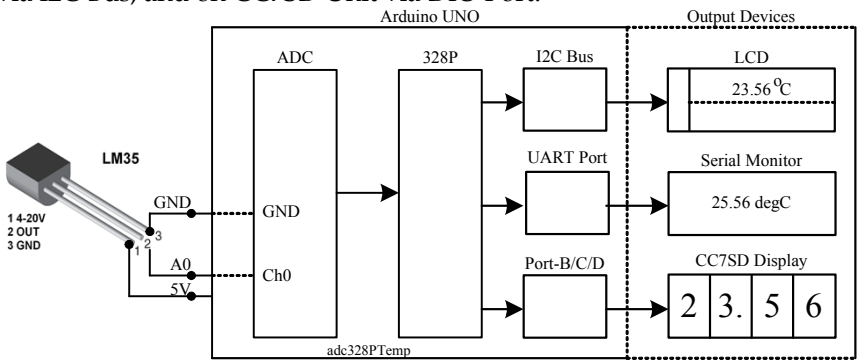

Figure-4.2: *Block Diagram of a LM35 sensor based Thermometer*

(2) LM35 Temperature Sensor: It is a factory calibrated analog type temperature sensor. It senses room temperature and produces 250 mV signals when room temperature is 25^0C; it produces 500 mV signals when the room temperature is 50^0C. Now, we have the following response points for the sensor:

(a) A(0.25V, 25^0C) //known point according to data sheets
(b) B(0.50V, 50^oC) //known point according to data sheets
(c) C(V_T, T^0C) //unknown point (the room temperature) that we want to know

The following response equation for the unknown temperature T can be easily deduces from the above there points: `T = 100*VT`. V_T is a DC voltage, and it can vary from 0 V to 1.50 V for a room temperature varying from 0^0C to 150^0C. Based on this fact, we may choose 1.1V INTERNAL as the reference voltage for the V_{REF}-pin of the ADC.

V_T enters into the ADC in analog form and comes out in the 10-bit digital form (ADCV, Section-4.1.6) with the following value: ADCV = (1023/1.1)*V_T. How?

(a) When input is 1.1 V, the ADC produces 1023 (all 1s = 11 1111 1111 = 1023)
(b) When input is V_T volt, the ADC produces 10-bit digital value (ADCV) = (1023/1.1)*V_T
(c) V_T = (1.1/1023)*ADCV. The value of ADCV can be known by executing this instruction: *analogRead(A0)*. Substituting value of V_T into this expression: T = 100*V_T, we get: `T = 100*(1.1/1023)*analogRead(A0).`

Precision of LM35 Sensor: Precision refers to number of digits that we want to keep after the decimal point. Because LM35 is an analog sensor, its output could be 250000 µV for 25^0C temperature. For what temperature, the sensor will produce 257654 µV signals? The answer is: 25.7654^0C. From this simple analysis, we may conclude that the equation `T = 100*(1.1/1023)*analogRead(A0)` holds both integer part and fractional part of the temperature. Now, we have the following commands:

(a)
```
int tempI = (int)100*(1.1/1023)*analogRead(A0);  // to keep only integer part of temperature
```
(b)
```
float tempF = (float)100*(1.1/1023.0)*analogRead(A0); //to keep both integer and fractional parts
```

(c) `Serial.print(tempI, DEC);  //will show integer part of temperature`

(d) `Serial.print(tempF, 2);  //will show float temperature with 2-digit after decimal point`

4.4 Sketch to show Room Temperature on Serial Monitor

```
void setup()
{
     Serial.begin(9600);
     analogRefernce(INTERNAL);//VREF pin is now connected to internal 1.1V reference; Full Scale = 1.1V
}

void loop()
{
     float tempF = (float)100*(1.1/1023.0)*analogRead(A0); //output signal of LM35 is with with Ch-0
     Serial.println(tempF, 2);     //shows decimal value of room temperature with 2-digit after point
     delay(1000);                  //1-sec time interval
}
```

4.5 Sketch to show Room Temperature on LCD

This sketch will show temperature reading of LM35 at 1-sec interval at top line of the LCD of Fig-4.2. Not that the LCD is an I2C Bus compatible, and it has been interfaced with UNO by I2C Controller.

```
#includcde<LiquidCrystal_I2C.h>          //include Library File to handle I2C Bus LCD
LiquidCrystal_I2C lcd(0x27, 16, 2);     //create object lcd with address 0x27, 16 characters, 2 line

void setup()
{
     Serial.begin(9600);
     analogRefernce(INTERNAL);//VREF pin is now connected to internal 1.1V reference; Full Scale = 1.1V

     //--next 3 lines are for I2C based LCD----
     lcd.init();              //calling library function/method to initialize lcd
     lcd.backlight();         //this function brings light at the back of LCD
     lcd.setCursor(0, 0);     //1st argument refers display position 0; 2ndarg = 0 refers TopLine
}

void loop()
{
     float tempF = (float)100*(1.1/1023.0)*analogRead(A0); //output signal of LM35 is with Ch-0
     lcd.print(tempF, 2);     //shows decimal value of room temperature with 2-digit after point
     delay(1000);             //1-sec time interval
}
```

4.6 Sketch to show Room Temperature on CC7SD Unit

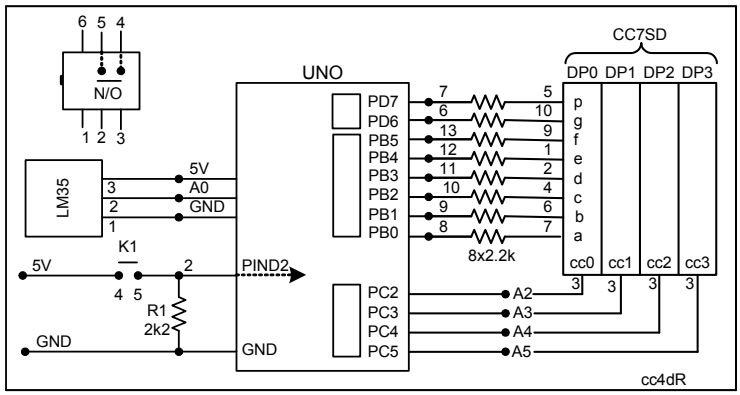

Figure-4.3: *Connection diagram among LM35, K1, UNO, and 4xCC7SD Unit*

(1) Sketch to show only the integer part of temperature at DP0-DP1 position of the display unit of Fig-4.3. The temperature acquisition interval is 1-sec, and the temperature measurement will begin when the push button K1 is just pressed.

(a) Acquire temperature:

```
int x = analogRead(A0);
```

(b) Derive the 2-digit integer part of the temperature:

```
int tempI = (int)100*(1.1/1023)*x;   //assume tempI = 23 (0x17)
```

(c) Isolate 3 from tempI variable, and save it in a variable named indexU.

```
byte indexU = tempI%10;  //indexU = 00000011 = 0x03
 tempI = temp1/10;            //tempI = 00000010 = 0x02
```

(d) Isolate 2 from tempI of Step-c, and save it in a variable named indexT.

```
byte indexT = tempI%10;       //indexT = 00000010 = 0x02
 tempI = temp1/10;            //tempI = 00000000 = 0x00
```

(e) Declare lupTable[10] for Digit-Vs-CCcode for 0 to 9.

```
 byte lupTable[10] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F};
```

(f) Consult LUT Table of Step-e; utilize indexU and indexT as array pointer to get the cc-codes for the unit-position digit and ten-position digit of temperatures. Save the cc-codes in the array byte ccArray[2]; where, ccArray[0] holds the cc-code of that digit that will appear at DP0-position of display unit.

```
ccArray[0] = lupTable[indexT];
ccArray[1] = lupTable[indexU];
```

(g) Transfer the contents of byte ccArray[2] onto DP0-DP1 positions of display unit.

```
void loop()
{
      PORTB = ccArray[0];          //bit-0 to bit-5 goes to segment-a to segment-f
      digitalWrite(6, bitRead(ccArray[0], 6));    //bit-6  goes to segment-g via PD6
      digitalWrite(7, bitRead(ccArray[0], 7));    //bit-7  goes to segment-p via PD7
      digitalWrite(A2, LOW);
      digitalWrite(A3, HIGH);
      delay(1);
      //-------------------------------------------------------------------------
      PORTB = ccArray[1];          //bit-0 to bit-5 goes to segment-a to segment-f
      digitalWrite(6, bitRead(ccArray[1], 6));    //bit-6  goes to segment-g via PD6
      digitalWrite(7, bitRead(ccArray[1], 7));    //bit-7  goes to segment-p via PD7
      digitalWrite(A2, HIGH);
      digitalWrite(A3, LOW);
      delay(1);
 }
```

4.4 Problems

1 The bit layout diagram for the ADMUX Register of the ADC Module is given below. Study the functions of the bits in consultation with data sheets and then answer to the questions:

ADC Multiplexer Selection Register ADMUX

Bit	7	6	5	4	3	2	1	0
	REFS1	REFS0	ADLAR		MUX3	MUX2	MUX1	MUX0
Access	R/W	R/W	R/W		R/W	R/W	R/W	R/W
Reset	0	0	0		0	0	0	0

(a) What will be open/close conditions of the switches K1 – K3 of Fig-4.1 when [REFS1:REFS0] = [1:0]?

(b) Write register level code using ADMUX to select Ch15 and V_{REF} at INTERNAL.

2 How much inaccurate temperature measurement would be there if we operate LM35 sensor using 5V for V_{REF} pin?

3 Create sketch to measure and display the Vcc supply voltage of the UNO Board.