Topic 3 : ADC



Problem 1 : An embedded system using ATMEGA32 working with 8Mhz. Write the C Program to The system interface with an analog input at channel ADC3, AREF=5V, left justified data, CLK/128 (using a variable resistor to make a voltage divider) the data will be displayed in PORTD ( LOW BYTE) . PORTC (HIGH BYTE) using Polling method ( or Interrupt method).

Problem 2 : Convert the C Code in Problem 1 into Assembly Code

**Problem 1: ADC Interface Using ATmega32 (Polling Method)**

#include <avr/io.h>

void adc\_init() {

// Set reference voltage to AVCC (5V)

ADMUX |= (1 << REFS0);

// Set left-adjusted result

ADMUX |= (1 << ADLAR);

// Set ADC clock prescaler to 128

ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);

// Enable ADC

ADCSRA |= (1 << ADEN);

}

uint16\_t read\_adc(uint8\_t channel) {

// Select ADC channel

ADMUX = (ADMUX & 0xF8) | (channel & 0x07);

// Start conversion

ADCSRA |= (1 << ADSC);

// Wait for conversion to complete

while (ADCSRA & (1 << ADSC));

// Read ADC result (16-bit)

return ADC;

}

int main() {

// Initialize ADC

adc\_init();

// Set PORTD and PORTC as output

DDRD = 0xFF;

DDRC = 0xFF;

while (1) {

// Read ADC value from channel ADC3

uint16\_t adc\_value = read\_adc(3);

// Split 16-bit value into low and high bytes

uint8\_t low\_byte = adc\_value & 0xFF;

uint8\_t high\_byte = (adc\_value >> 8) & 0xFF;

// Display data on PORTD and PORTC

PORTD = low\_byte;

PORTC = high\_byte;

}

return 0;

}

**Problem 2: Assembly Code for Problem 1**

.include "m328pdef.inc"

.org 0x0000

rjmp main

main:

; Initialize ADC

ldi r16, (1 << REFS0) ; Set reference voltage to AVCC (5V)

out ADMUX, r16

ldi r16, (1 << ADLAR) ; Set left-adjusted result

out ADMUX, r16

ldi r16, (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0) ; Set ADC clock prescaler to 128

out ADCSRA, r16

ldi r16, (1 << ADEN) ; Enable ADC

out ADCSRA, r16

; Set PORTD and PORTC as output

ldi r16, 0xFF

out DDRD, r16

out DDRC, r16

loop:

; Read ADC value from channel ADC3

ldi r17, 3

call read\_adc

; Split 16-bit value into low and high bytes

mov r18, r17

andi r18, 0xFF ; low\_byte

mov r19, r17

swap r19

andi r19, 0xFF ; high\_byte

; Display data on PORTD and PORTC

out PORTD, r18

out PORTC, r19

rjmp loop

read\_adc:

; Select ADC channel

andi r17, 0x07

ldi r16, (1 << ADLAR) | (1 << ADEN) | (1 << ADSC) ; Start conversion

out ADCSRA, r16

loop\_until\_bit\_is\_clear ADCSRA, ADSC ; Wait for conversion to complete

in r17, ADC ; Read ADC result (16-bit)

ret