ECED 3204 Microprocessor

Assignment #8 Reference Solution

http://www.jasongu.org/3204/assignments.html

Assignment #8 contains the following problems:

E16.1 Design a circuit that can scale the voltage from the range of 0 mV ~ 100 mV to the range of 0 V ~ 5 V.

Solution:
The circuit in Figure 16.6 should be used to do the job:

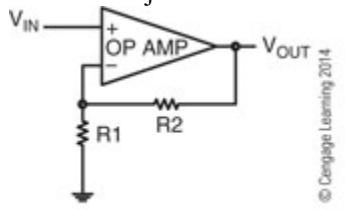


Figure 16.6 ■ A voltage scaler

5 V / 0.1 V = 50, ∴R2/R1 = 49. We will choose 49 k□ and 1 k□ for R2 and R1, respectively. Since there is no 49 k□ resistor, it can be broken down to 47k□ and 2 k□ resistors.

E16.6 Suppose that there is a 12-bit A/D converter with V_{RL} and V_{RH} set to 1.5 V and 3.5 V respectively. Find the corresponding voltage values for the A/D conversion results of 80, 180, 480, 640, 960, 1600, 2048, 3200, and 4000, respectively.

Solution:
V(80) = 1.5 + 80 x 2 / 4095 = 1.54 V
V(180) = 1.5 + 180 x 2 /4095 = 1.59 V
V(480) = 1.5 + 480 x 2 / 4095 = 1.73 V
V(640) = 1.5 + 640 x 2 / 4095 = 1.81 V
V(960) = 1.5 + 960 x 2 / 4095 = 1.97 V
V(1600) = 1.5 + 1600 x 2 / 4095 = 2.28 V
V(2048) = 1.5 + 2048 x 2 / 4095 = 2.5 V
V(3200) = 1.5 + 3200 x 2 / 4095 = 3.06 V
V(4000) = 1.5 + 4000 x 2 / 4096 = 3.45 V

E16.7 Write a sequence of AVR instructions and C statements to configure the AVR MEGA2560 ADC to operate with the following settings:

- Select differential mode input
- Select ADC1 and ADC0 as positive and negative inputs with gain set to 10
- Disable auto-triggering
- Set ADC clock prescaler to 128 assuming that the MEGA2560 uses a 16-MHz crystal oscillator to generate its system clock
- Select AVCC as its reference voltage
- Result right-justified

- Disable digital input buffers of the unused analog inputs
- Disable ADC interrupt

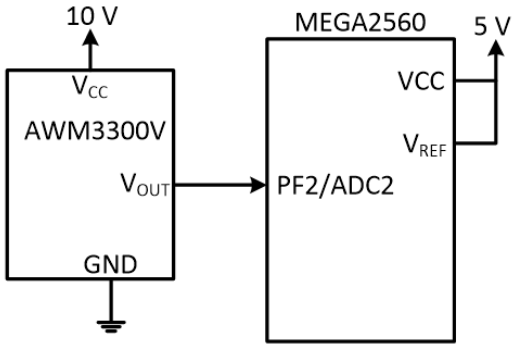
Solution:
The assembly subroutine that performs the specified setting is as follows:

```
initADC:    ldi     r20,0x87        ; enable ADC, set ADC clock prescaler to 128, disable
            sts     ADCSRA,r20      ; auto triggering, disable ADC interrupt
            clr     r20
            sts     ADCSRB,r20      ; ADC result right justified
            ldi     r20,0x49        ; set ADC1 and ADC0 pins as pos and neg input
            sts     ADMUX,r20       ;     "
            lds     r20,DDRF        ; configure PF0/ADC0, PF1/ADC1 pins for input
            andi    r20,0xFC        ;     “
            sts     DDRF,r20        ;     “
            ldi     r20,0xFC        ; disable digital input buffers of the unused analog input
            sts     DIDR0,r20       ;     "
            ldi     r20,0xFF        ;     "
            sts     DIDR1,r20       ;     "
            ret
```

The C function that performs the same setting is as follows:

```
void initADC(void)
{
    ADCSRA    = 0x87; // set ADC clock prescaler to 128, disable ADC interrupt
    ADCSRB    = 0;    // ADC result right justified
    ADMUX     = 0x49; // set ADC1 & ADC0 as positive and negative inputs
    DDRF      &= 0xFC; // configure PF0/ADC0, PF1/ADC1 pins for input
    DIDR0     = 0xFC; // disable digital input buffers of the unused analog input
    DIDR1     = 0xFF;       // pins
}
```

E16.16 The Microbridge AWM3300V is a mass airflow sensor manufactured by Honeywell. The block diagram of the AWM3300V is shown in Figure E16.16. It is designed to measure the airflow. Its applications include air-conditioning, medical ventilation/anesthesia control, gas analyzers, gas metering, fume cabinets, and process control. The AWM3300V operates on a single 10 V ± 10 mV power supply. The sensor output (V_{OUT}) corresponding to the airflow rate of 0 ~ 1.0 L/min is 1.0 V to 5.0 V. The AWM3300V can operate in the temperature range from -25 to 85ºC. It takes about 3 ms for the output voltage to settle after power up. Design a circuit to measure and display the mass airflow using the AWM3300V. Write a program to configure the MEGA2560 ADC module, start the A/D conversion, and display the mass airflow in an LCD display. Update the display every 200 ms.



Solution:
If we connect the AWM3300V output directly to the MEGA2560 ADC module, then the conversion result will be in the range from 205 to 1023. The conversion result can be translated back to massflow using the following equation:

Massflow = (ADC result – 205) × 1000 ÷ 818

The circuit connection is shown in Figure E16.16.

The C program that measures the massflow once every second is as follows:

```
#include <avr\io.h>
#include "usartUtil_mega.h"
#include "delays_mega2560.h"
unsigned char massFlow[10]; // output buffer
unsigned char *msg1 = "massFlow = ";
void initADC(void);
void main (void)
{
	unsigned int temp, tmp;
	initADC();
	initUSART0();
	initADC();
	massFlow[4] = 'm'; // space character
	massFlow[5] = 'l';
	massFlow[6] = 0; // NULL character
	while(1){
		ADCSRA	|= (1 << ADSC); // start A/D conversion
		while(!(ADCSRA & (1<<ADIF))); // wait until A/D conversion is completed
		ADCSRA |= (1<<ADIF); // clear ADIF flag
		temp = ADC - 205; // subtract offset corresponding to massflow 0 ml/s
		tmp	= (unsigned int)(((long)temp)*1000/818); // convert to massflow
		massFlow[3] = tmp % 10 + 0x30; // ASCII code of one's digit
		tmp /= 10;
		if(tmp == 0){ // If upper three digits are all zero
			massFlow[2] = 0x20;
			massFlow[1] = 0x20;
			massFlow[0] = 0x20;
		} else { // upper three digits are not all zero, separate ten's digit
			massFlow[2] = tmp % 10 + 0x30; // ASCII code of ten's digit
			tmp /= 10;
		}
		if(tmp == 0){// If upper two digits are all zero
			massFlow[1] = 0x20;
			massFlow[0] = 0x20;
		} else {
			massFlow[1] = tmp % 10 + 0x30; // ASCII code of hundred's digit
			tmp /= 10;
		}
		if(tmp == 0) // thousabd's digit is zero?
			massFlow[0] = 0x00;
		else
			massFlow[0] = 0x31;
```

```
            newline();
            putsUSART0(msg1);
            putsUSART0(massFlow);
            delayby1s(1);
    }
}

void initADC(void)
{
    ADCSRA    = 0x87;       // enable ADC, disable auto trigger, set ADC clock to 125 kHz
    ADMUX     = 0x42;       // use AREF as reference voltage, select ADC2 pin, result right-justified
    ADCSRB    = 0;          // single-ended channel
    DIDR0     = 0xFB;       // disable digital input buffers for all except ADC2 input
    DIDR1     = 0xFF;       //        "
    DDRF      |= 0xFB;      // configure PF2/ADC2 pin for input
}
```