**Digital Voltmeter**

**1. Introduction**

**2. Experiments**

**2.1 External D/A converter**

1. **General Overview**
2. **Register Settings**
3. **The C Code**

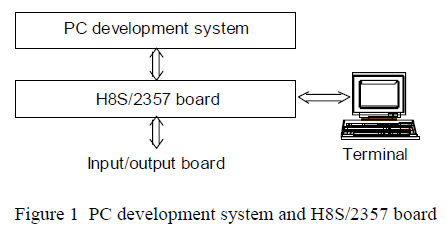
**2.2 Internal A/D converter**

1. **General Overview**
2. **The C Code**

**3. Conclusion**

1. **Introduction**

The assignment for this lab mainly consists of two parts. Both parts demand the creation of a C program: the first one – for an external D/A converter, the second one – for an internal A/D converter. The Figure 1 shows the appropriate PC development system and H8S/2357 board for the implementation of the created code.



In the lab the appropriate connection between the components of the digital voltmeter and the input/output board should be created. Furthermore, the code that was written as a part of the preparation for the lab should be implemented via using the editor, compiler, and linker of the PC system. That’s how the control software for the programs is developed. Finally, the control software should be transferred to the H8S/2357 board for test.

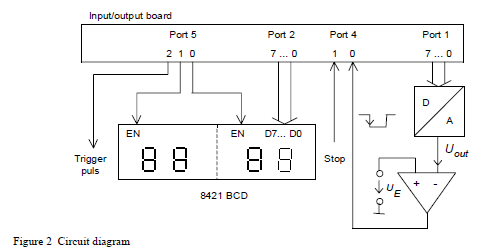
1. **Experiments**

**2.1 External D/A converter**

**1. General overview**

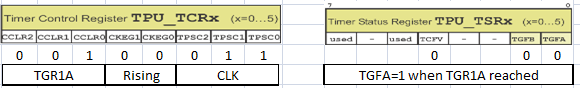
A digital-to-analog converter (D/) is a device which is used to handle the conversion of a digital value to an analog signal. The digital value is usually represented by a binary code, while an analog signal can be current, voltage, or electric charge.

The transfer characteristics of the internal D/A converter were given in the assignment. The corresponding circuit diagram is shown on Figure 2. The method that was used in order to develop a C program for such a converter is called weighting conversion method. This method is based on the following: first, an output is a binary number with only MSB set. Then the comparison of the corresponding output voltage Uout with the input voltage Ue is done. If Uout is less than Ue, the voltage level at PG(0) is set to high level. If Uout is more than Ue, the comparator puts out a low level. So, the decision whether the MSB should be cleared or not is determined. The sequence of these steps is holded for all remaining bits and as a result we get a binary value which corresponds to the input voltage Ue. Afterwards, the voltage is being output. Ports(J) and(E) are used for displaying three digits of this output. When the change of the digital input occurs, it takes some time for a comparator to come to a stable mode. In our program we should consider the delay for the comparator which is equal to 30µs.

****

**2. Register Settings**

For the function wait () in our C code we used the following register settings:



**3. The C Code**

#include <mpp1.h>

#include <mpp82357.h>

void wait\_for(void)

{

TPU\_TCR1=0x23; //counter restarts with TGR1A, rising edge, CLK/64

TPU\_TCNT1=0x0000; //initialize the counter

TPU\_TGR1A=(10-1); //for 3.472 µs per tick for CLK/64, 9 ticks will give ~ 30 µs

//(31.248 µs)

TPU\_TSTR=0x02; //start timer for Channel 1

while((TPU\_TSR1&0x01)==0); //wait until TGFA is reached

TPU\_TSR1&= 0xFE; //clear the TGFA

TPU\_TSTR=0x00; // stop the counter

}

void display(void) // the voltage is displayed

{

P5DR |= (1<<0); // refer to EN1

P2DR = display2; // write the value to EN1

wait\_for(); // wait for 31.248 µs

P5DR &= ~(1<<0); // EN1 is disabled

P5DR |= (1<<1); // refer to EN0

P2DR = display1; // write the value to EN0

wait\_for(); // wait for 31.248 µs

P5DR &= ~(1<<1); // EN0 is disabled

}

int main(void)

{

int temp;

int counter;

float out\_voltage;

unsigned char display1;

unsigned char display2;

P1DDR = 0xFF; // PORT1 (7…0) is set to output

P2DDR = 0xFF; // PORT2 (7…0) is set to output

P5DDR = 0x07; // PORT5 (2…0) is set to output

P5DR = 0x00; // write the initial value for PORT5

do

{

for(counter = 7, P1DR = 0; counter >= 0; counter--) // loop for D/A conversion

{

P1DR |= (1<<counter); // the current bit of PORT1 is set now

wait\_for(); // wait for 31.248 µs

if(!(PORT4&(1<<0))) // out\_voltage (Uout) > ref\_voltage (Ue)

P1DR &= ~(1<<counter); // the bit is cleared now

}

display(); //call display function

}

while(PORT4&(1<<1)); // do while the stop-button wasn’t pressed

return 0;

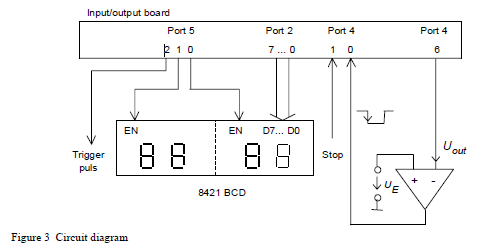
}

**2.2 Internal A/D converter**

**1. General overview**

An analog-to-digital converter (A/D) is a device which is used to handle the conversion of an analog signal such as voltage, current, or electric charge to a digital value. This digital value is usually represented by a binary code.

The table containing the transfer characteristics of the internal A/D converter was given in the assignment. The result of the A/D conversion according to our assignment should be a transformation of the analog voltage Ue into a corresponding binary number. This binary number should be proportional to Ue. The voltage has to be output. Ports 2 and 5 are used for displaying four digits of this output. The corresponding circuit diagram is shown on Figure 3.



**2. The C Code**

#include <mpp1.h>

#include <mpp82357.h>

void wait\_for(void)

{

TPU\_TCR1=0x23; //counter restarts with TGR1A, rising edge, CLK/64

TPU\_TCNT1=0x0000; //initialize the counter

TPU\_TGR1A=(10-1); //for 3.472 µs per tick for CLK/64, 9 ticks will give ~ 30 µs

// (31.248 µs)

TPU\_TSTR=0x02; //start timer for Channel 1

while((TPU\_TSR1&0x01)==0); //wait until TGFA is reached

TPU\_TSR1&= 0xFE; //clear the TGFA

TPU\_TSTR=0x00; // stop the counter

}

void display(void) // the voltage is displayed

{

// display the voltage

P5DR |= (1<<0); // refer to EN1

P2DR = display2;// write the value to EN1

wait\_for(); // wait for 31.248 µs

P5DR &= ~(1<<0); // EN1 is disabled

P5DR |= (1<<1); // refer to EN0

P2DR = display1; // write the value to EN0

wait\_for(); // wait for 31.248 µs

P5DR &= ~(1<<1); // EN0 is disabled

}

int main(void)

{

int temp;

float ref\_voltage;

unsigned short an0;

unsigned char display1;

unsigned char display2;

P2DDR = 0xFF; // PORT2 (7…0) is set to output

P5DDR = 0x03; // PORT5 (1…0) is set to output

P5DR = 0x00; // write the initial value for PORT5

ADCSR = 0x00; // single mode, interrupt request disabled, conversion time=266 states,

// channel AN0 is used

do

{

ADCSR |= (1<<ADST); // A/D conversion started

while(!(ADCSR&(1<<ADF))); // wait while the A/D conversion is in process

ADCSR &= ~(1<<ADF); // ADF flag is cleared

an0 = ADDRAH << 2; // the result of A/D conversion is read

an0 |= ADDRAL >> 6; // the result of A/D conversion is read

display();

}

while(PORT4&(1<<1)); // do while the stop-button wasn’t pressed

return 0;

}

1. **Conclusion**

The aim of this lab was to understand the principle of working for both D/A and A/D conversions. For both parts of the assignments the appropriate methods for conversion were applied.

We managed to create a correctly functioning code during the preparation and in the lab we found out that this codes are also working properly on the PC system for development and H8S/2357 board.