Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Рязанский государственный радиотехнический университет имени В.Ф. Уткина»

Кафедра «Электронные вычислительные машины»

ОТЧЕТ

По лабораторная работа №4

**Особенности выполнения арифметических**

**и логических операций в RISC процессорах**

По дисциплине «ЭВМ и периферийные устройства»

Выполнили:

Проверили:

Рязань 20

**Цель:** познакомиться с понятием «выводы общего назначения», получить

практические навыки по устройству и настройке параллельного порта, и выводов общего назначения в МК AVR, закрепить на практике навыки программирования на языке Assembler и C.

**Задание 1.** Измените код из примера 1 таким образом, чтобы при переполнении значения регистра r1 счетчик менял направление счета с +1 на -1, а при займе – с -1 на +1. Проанализируйте работу программы в отладчике.

Анализатор подключите к выводам согласно рис. 4.2а. загрузите программу в

контроллер. Проанализируйте поведение сигналов в программе Logic.

Код задания 1 на Assembler:

start:

ldi r16, 0xFF

ldi r17, 0x00

mov r1, r17

ldi r18, 0x01//+-1

ldi r19, 0b11111110//маска для смены знака

ldi r20, 0x00

out DDRD,r16

p:

out PORTD, r1

add r17, r18// r1 заменим на r17, потому что так лучше//\*/

mov r1, r17

BRNE m1//если переполнение или обнуление меняем знак у 1

eor r18, r19

CPI r18, -1

BRNE m1

ldi r17, 0xfe//если начинаем вычитание - установить в счётчик 0xfe

mov r1, r17

m1:

rjmp p

Фрагменты трассы выполнения программы - таблицы 1 и 2.

Таблица 1. максимум счётчика. Начало таблицы

|  |  |  |  |
| --- | --- | --- | --- |
| № строки | команда | действие | результат |
| 8 | p: out PORTD, r17 | вывести r17 в порт d | PORTD=0xfd |
| 9 | add r17, r18 | прибавить к r17 r18 | r17=0xfe Z=0 |
| 10 | BRNE m1 | если Z=0 перейти к m1 |  |
| 15 | m1: rjmp p |  |  |
| 8 | p: out PORTD, r17 | вывести r17 в порт d | PORTD=0xfe |
| 9 | add r17, r18 | прибавить к r17 r18 | r17=0xff Z=0 |
| 10 | BRNE m1 | если Z=0 перейти к m1 |  |
| 15 | m1: rjmp p |  |  |
| 8 | p: out PORTD, r17 | вывести r17 в порт d | PORTD=0xff |
| 9 | add r17, r18 | прибавить к r17 r18 | r17=0x00 Z=0 |
| 10 | BRNE m1 | если Z=0 перейти к m1 |  |

Таблица 1. максимум счётчика. Завершение таблицы

|  |  |  |  |
| --- | --- | --- | --- |
| 11 | eor r18, r19 | исключающее или r18 и числа 0b11111110 | r18=0xff |
| 12 | CPI r18, -1 | сравниваем r18 и -1 | Z=1 |
| 13 | BRNE m1 | если Z=0 перейти к m1 |  |
| 14 | ldi r17, 0xfe | присваеваем r17 значение 0xfe | r17=0xfe |
| 15 | m1: rjmp p |  |  |
| 8 | p: out PORTD, r17 | вывести r17 в порт d | PORTD=0xfe |
| 9 | add r17, r18 | прибавить к r17 r18 | r17=0xfd Z=0 |
| 10 | BRNE m1 | если Z=0 перейти к m1 |  |
| 15 | m1: rjmp p |  |  |

Таблица 2. минимум счётчика

|  |  |  |  |
| --- | --- | --- | --- |
| № строки | команда | действие | результат |
| 8 | p: out PORTD, r17 | вывести r17 в порт d | PORTD=0x02 |
| 9 | add r17, r18 | прибавить к r17 r18 | r17=0x01 Z=0 |
| 10 | BRNE m1 | если Z=0 перейти к m1 |  |
| 15 | m1: rjmp p |  |  |
| 8 | p: out PORTD, r17 | вывести r17 в порт d | PORTD=0x01 |
| 9 | add r17, r18 | прибавить к r17 r18 | r17=0x00 Z=1 |
| 10 | BRNE m1 | если Z=0 перейти к m1 |  |
| 11 | eor r18, r19 | исключающее или r18 и числа 0b11111110 | r18=0x01 |
| 12 | CPI r18, -1 | сравниваем r18 и -1 | Z=0 |
| 13 | BRNE m1 | если Z=0 перейти к m1 |  |
| 15 | m1: rjmp p |  |  |
| 8 | p: out PORTD, r17 | вывести r17 в порт d | PORTD=0x00 |
| 9 | add r17, r18 | прибавить к r17 r18 | r17=0x01 Z=0 |
| 10 | BRNE m1 | если Z=0 перейти к m1 |  |
| 15 | m1: rjmp p |  |  |
| 8 | p: out PORTD, r17 | вывести r17 в порт d | PORTD=0x01 |
| 9 | add r17, r18 | прибавить к r17 r18 | r17=0x02 Z=0 |
| 10 | BRNE m1 | если Z=0 перейти к m1 |  |
| 15 | m1: rjmp p |  |  |

Анализ сигналов в Logic - рисунок 1 (рисунки инвертированы для удоства прочтения в печатном виде)

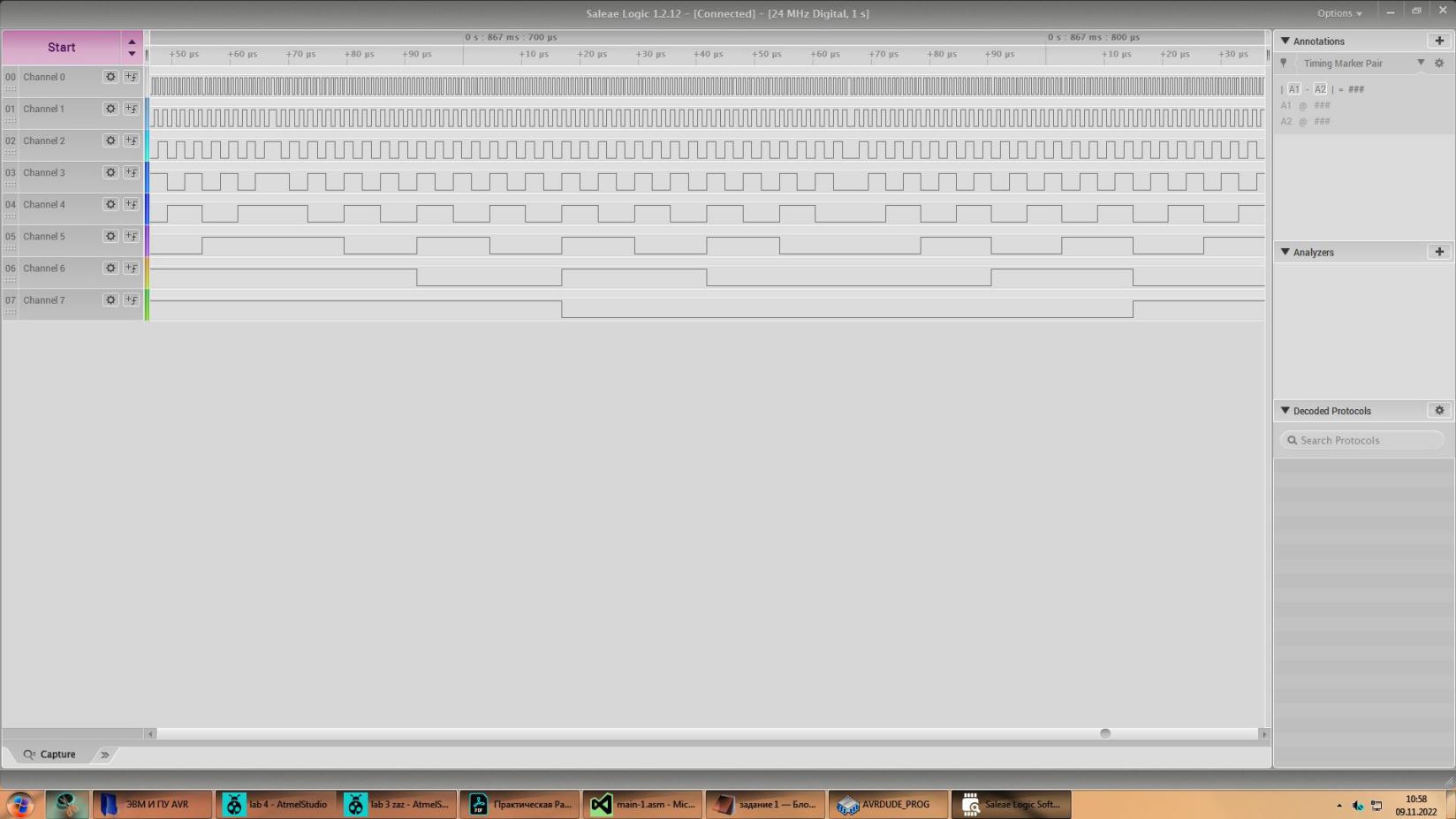


Рисунок 1 — выполнение задания 1 на Logic

В тот момент, когда у графиков сигналов на всех выводах порта D наблюдается максимум, счётчик достигает максимума. Аналогично с минимумом. На рисунке 1 видно и максимум и минимум счётчика.

**Задание 2.** Измените код программы из задания 1 таким образом, чтобы на

вывод работали только выводы PORTD6-PORTD7. Используя логический

анализатор, докажите корректность работы программы.

Код задания 2 на Assembler:

start:

ldi r16, 0xc0

ldi r17, 0x00

mov r1, r17

ldi r18, 0x01//+-1

ldi r19, 0b11111110//маска

out DDRD,r16

p:

out PORTD, r17

add r17, r18// r1 заменим на r17, потому что так лучше//\*/

mov r1, r17

BRNE m1

eor r18, r19

CPI r18, -1

BRNE m1

ldi r17, 0xfe

mov r1, r17

m1:

rjmp p

Выполнение программы в Logic - рисунок 2.

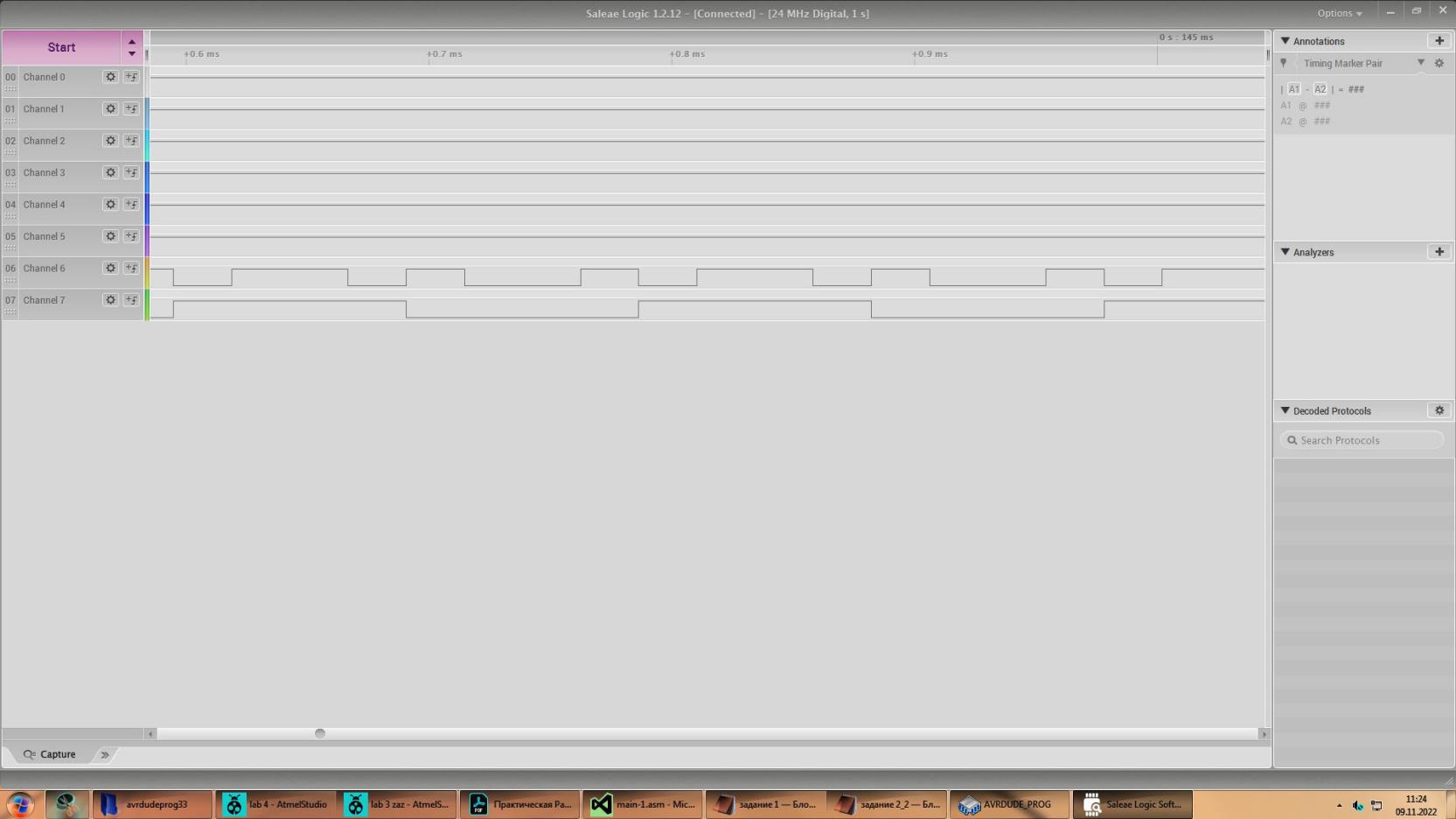


Рисунок 2 — выполнение задания 2 на Logic

На рисунке чётко видно смену знака изменения счётчика, так же по шкале измерения времени видно что период изменения каналов 7 и 8 одинаковый на рисунках 1, 2 и 3, следовательно программа работает корректно.

Примечание: код в котором число выводится целиком через PORTD6-PORTD7 смотри в приложении А.

**Задание 3.** Измените код программы таким образом, чтобы частота мигания светодиода изменилась по следующему правилу: Clk/16.

Код задания 3 на C:

#include <avr/io.h>

uint32\_t x=0;

int flag=0;

int main(void) {

DDRB = 0x20;

while (1)

{

if (x==800000\*16)//умножаем период на 16

{if (flag==0)

{

PORTB=0x20; flag=1;

}

else

{

PORTB=0x00; flag=0;

}

x=0;}

x++;

}}

Частота мигания светодиода уменьшилась в 16 раз.

**Задание 4.** Разработайте программу, которая по нажатию на кнопку 1

увеличивала бы, а по нажатию на кнопку 2 – уменьшала состояние регистра R0 на 1. Значение регистра R0 должно выводиться в порт D. Кнопки подключите к выводам (1 – PB0, 2 – PB1). Выполните сборку программы. Проанализируйте её в отладчике, имитируя нажатия на кнопки. Загрузите программу в МК, проанализируйте работу программы с помощью

логического анализатора. Анализатор подключите согласно рисунку 4.2.

Код задания 4 на C:

#include <avr/io.h>

int main(void) {

short int x;

DDRD =0xff;

PORTB |= 0x03;

while (1)

{

if((PINB&0x01)==0x00)

x++;

if((PINB&0x02)==0x00)

x--;

PORTD=x;

} }

Выполнение программы в Logic - рисунки 3,4.

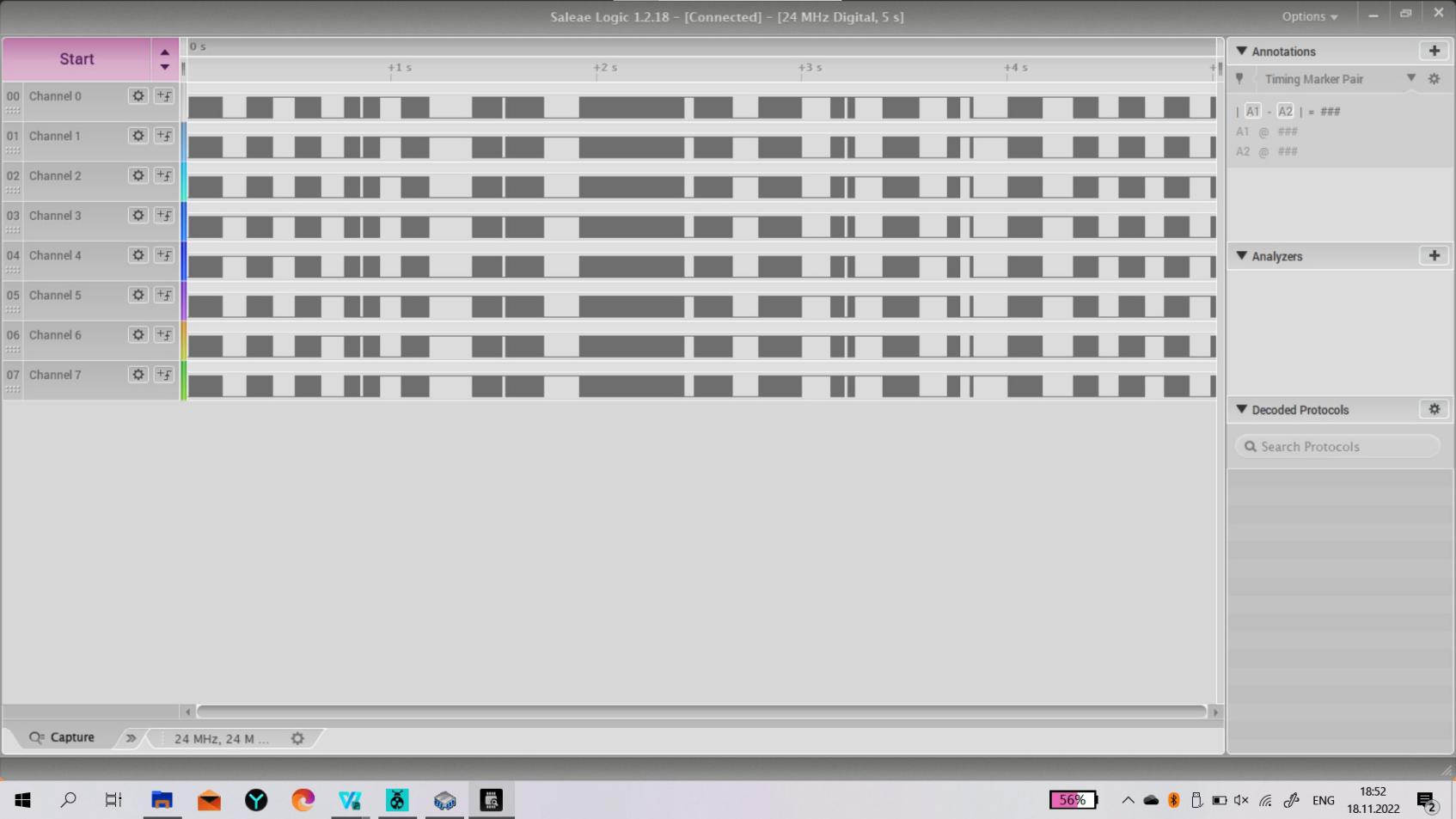


Рисунок 3 — выполнение задания 4 на Logic. 1

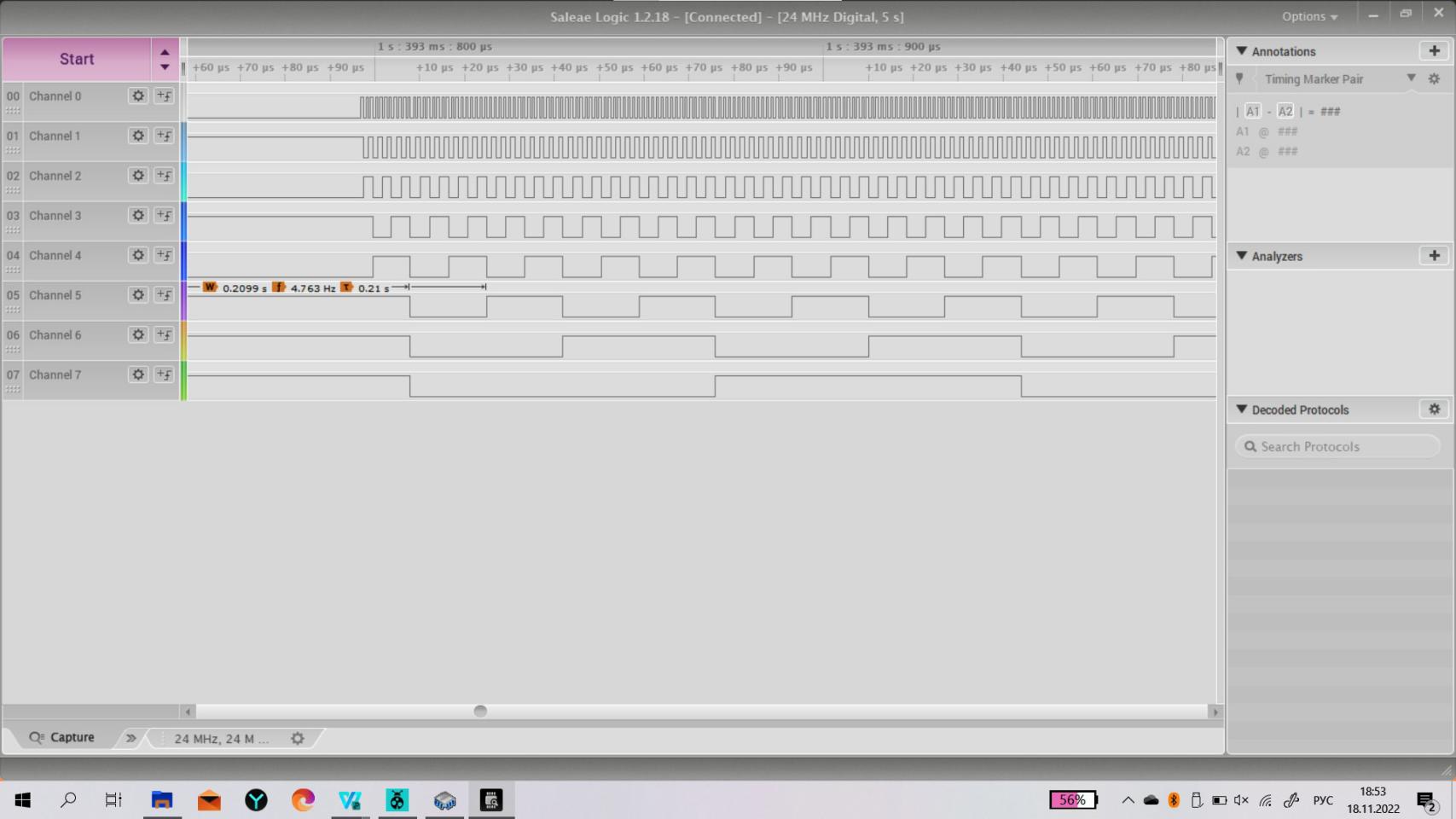


Рисунок 4 — выполнение задания 4 на Logic. 2

На рисунке 3 видно промежутки времени в которые были нажаты кнопки, рисунок 4 на более крупном масштабе и на нем видно изменение счётчика.

**Задание 5.** Разработайте программу на языке C, которая обеспечивала бы управление светодиодами (RX – PD0, TX – PD1, L – PB5) с помощью кнопок (кнопки подключите к выводам PD2, PD3, PD4). Нажатие на кнопку изменяет состояние светодиода на противоположное, после чего оно сохраняется. Выполните сборку программы. Проанализируйте её в отладчике, имитируя нажатия на кнопки. Подключите кнопки к плате в соответствии с заданием. Покажите подключение преподавателю. Загрузите программу в контроллер. Проанализируйте работу системы.

Код задания 5 на C:

#include <avr/io.h>

#include <stdbool.h>

int main(void) {

short int x;

\_Bool rx, tx, l;//хранит информацию о состоянии светодиода

DDRD =0x03;

DDRB =0x20;

PORTD |= 0x1c;

rx=true;

tx=true;

l=true;

while (1)

{

if((PIND&0x04)==0x00){

rx = ! rx;

if (rx)

PORTD|=0x01;

else

PORTD&=0xfe;

}

if((PIND&0x08)==0x00){

tx = ! tx;

if (tx)

PORTD|=0x02;

else

PORTD&=0xfd;

}

if((PIND&0x10)==0x00){

l = ! l;

if (l)

PORTB|=0x20;

else

PORTB&=~0x20;

}

} }

Результат выполнения программы в Logic - рисунки 5,6.

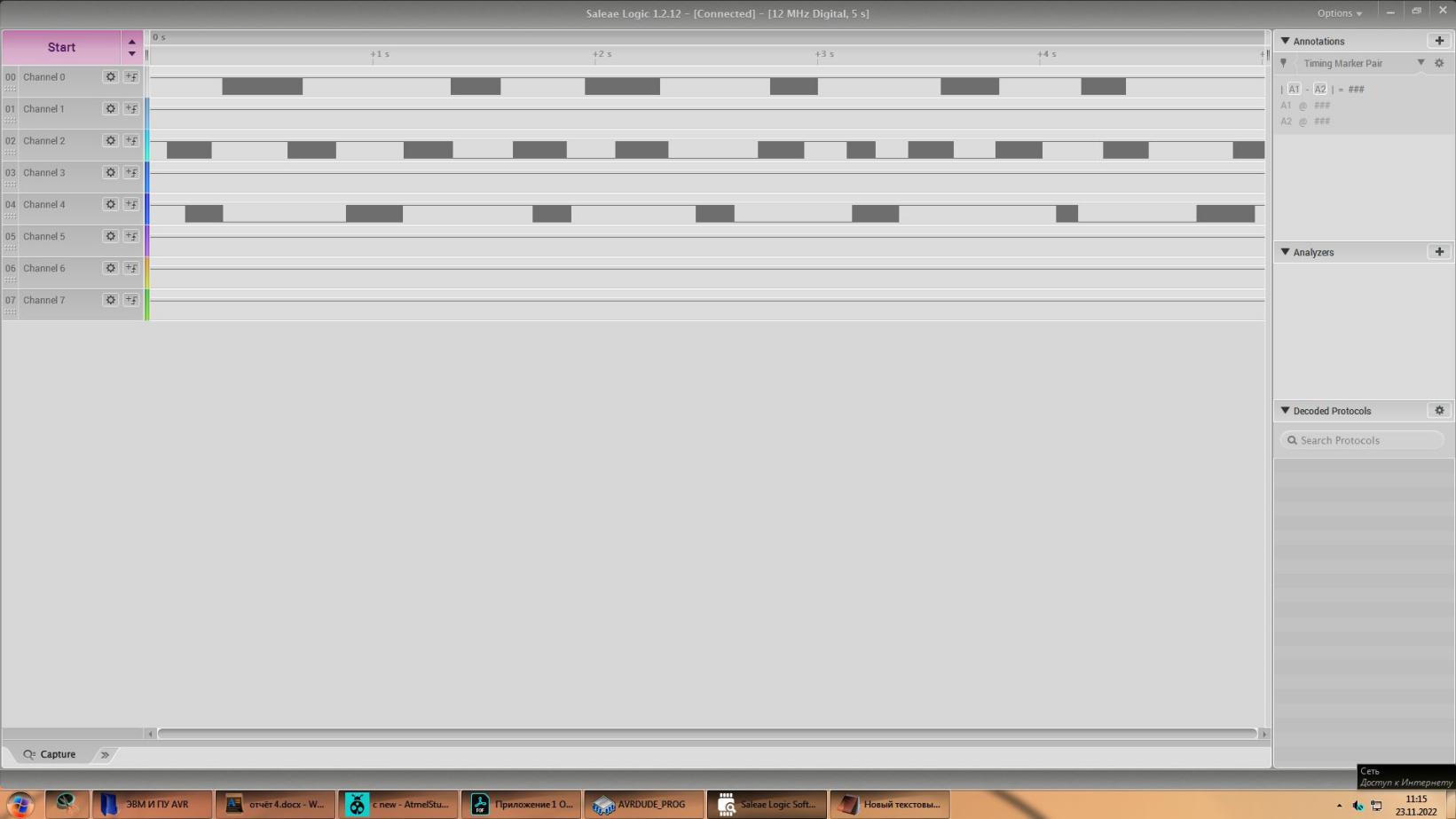


Рисунок 5 — выполнение задания 5 на Logic. 1

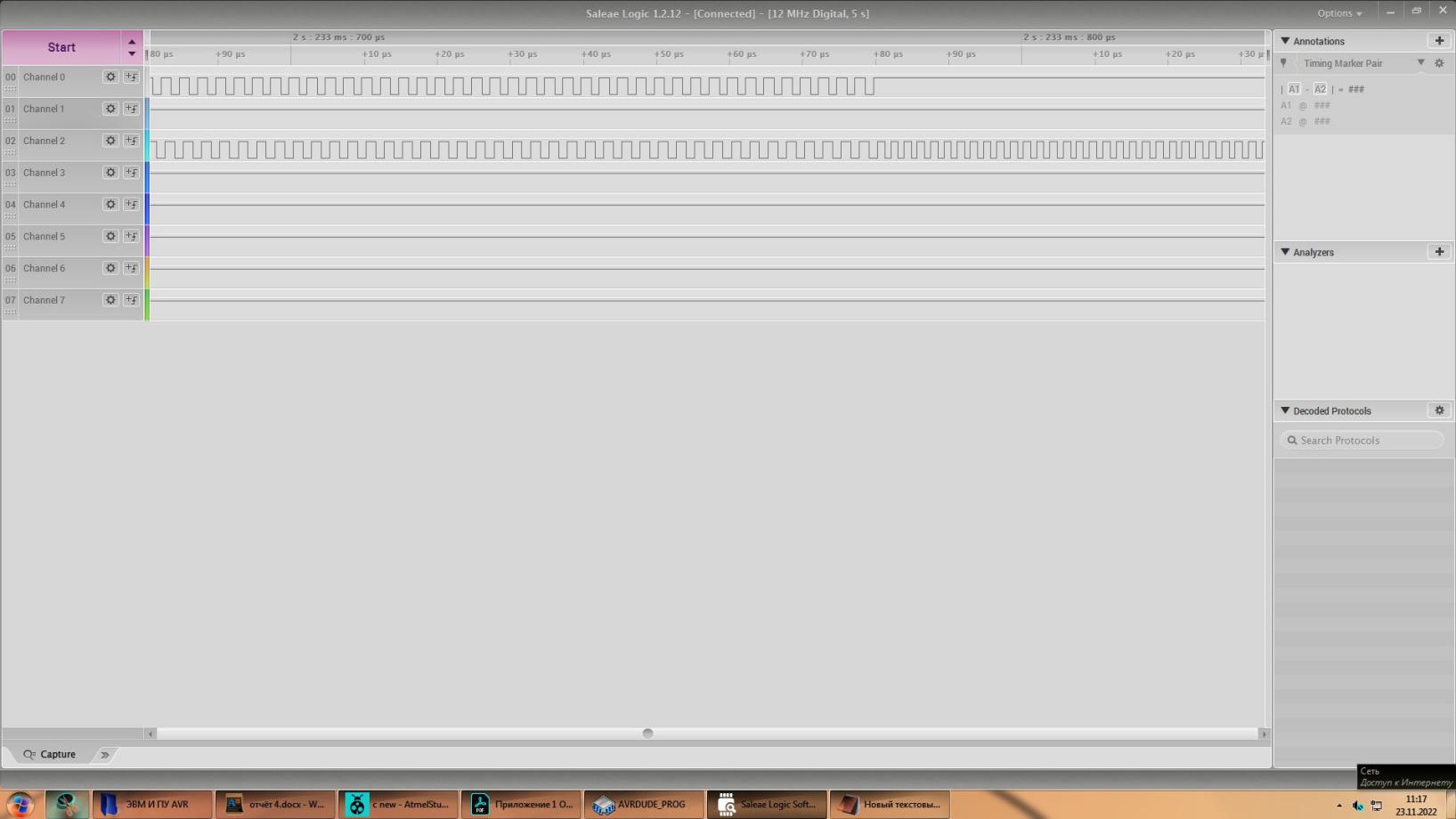


Рисунок 6 — выполнение задания 5 на Logic. 2

На рисунке 5 видны промежутки времени нажатия на кнопки, на рисунке 6 - изменение состояния светодиодов во время нажатия на кнопки.

При нажатии на кнопку 1 меняется состояние светодиода RX, на кнопку 2 - TX, на кнопку 3 - L.

**Вывод:** ознакомились с понятием «выводы общего назначения», получили практические навыки по устройству и настройке параллельного порта, и выводов общего назначения в МК AVR, закрепили на практике навыки программирования на языке Assembler и C.

**Приложение А**

Код программы к заданию 2, в котором всё число выводится через 2 старших бита порта в 2 вариантах

|  |  |
| --- | --- |
| Вариант 1. со сдвигами | Вариант 2. с масками |
| start:  ldi r16, 0xc0  ldi r17, 0x00  ldi r18, 0x01//+-1  ldi r19, 0b11111110//маска  ldi r20, 0x00  out DDRD,r16  p:  ldi r21, 0x03//01 биты  and r21, r17  LSL r21  LSL r21  LSL r21  LSL r21  LSL r21  LSL r21  ldi r22, 0x0c//23 биты  and r22, r17  LSL r22  LSL r22  LSL r22  LSL r22  ldi r23, 0x30//34 биты  and r23, r17  LSL r23  LSL r23  ldi r24, 0xc0//67 биты  and r24, r17  out PORTD, r24  out PORTD, r23  out PORTD, r22  out PORTD, r21  add r17, r18// r1 заменим на r17, потому что так лучше//\*/  BRNE m1  eor r18, r19  CPI r18, -1  BRNE m1  ldi r17, 0xfe  m1:  rjmp p | start:  ldi r16, 0xc0  ldi r17, 0x00  ldi r18, 0x01//+-1  ldi r19, 0b11111110//маска  ldi r20, 0x00  ldi r25, 0x40  ldi r26, 0x20  ldi r27, 0x10  out DDRD,r16  p:  ldi r21, 0x03//01 биты  and r21, r17  mul r21, r25  mov r21, r0  ldi r22, 0x0c//23 биты  and r22, r17  mul r22, r26  mov r22, r0  ldi r23, 0x30//34 биты  and r23, r17  mul r23, r27  mov r23, r0  ldi r24, 0xc0//67 биты  and r24, r17  out PORTD, r24  out PORTD, r23  out PORTD, r22  out PORTD, r21  add r17, r18// r1 заменим на r17, потому что так лучше//\*/  BRNE m1  eor r18, r19  CPI r18, -1  BRNE m1  ldi r17, 0xfe  m1:  rjmp p |

Выполнение кодов на Logic эквивалентны и имеют вид, представленный на рисунках 7,8.

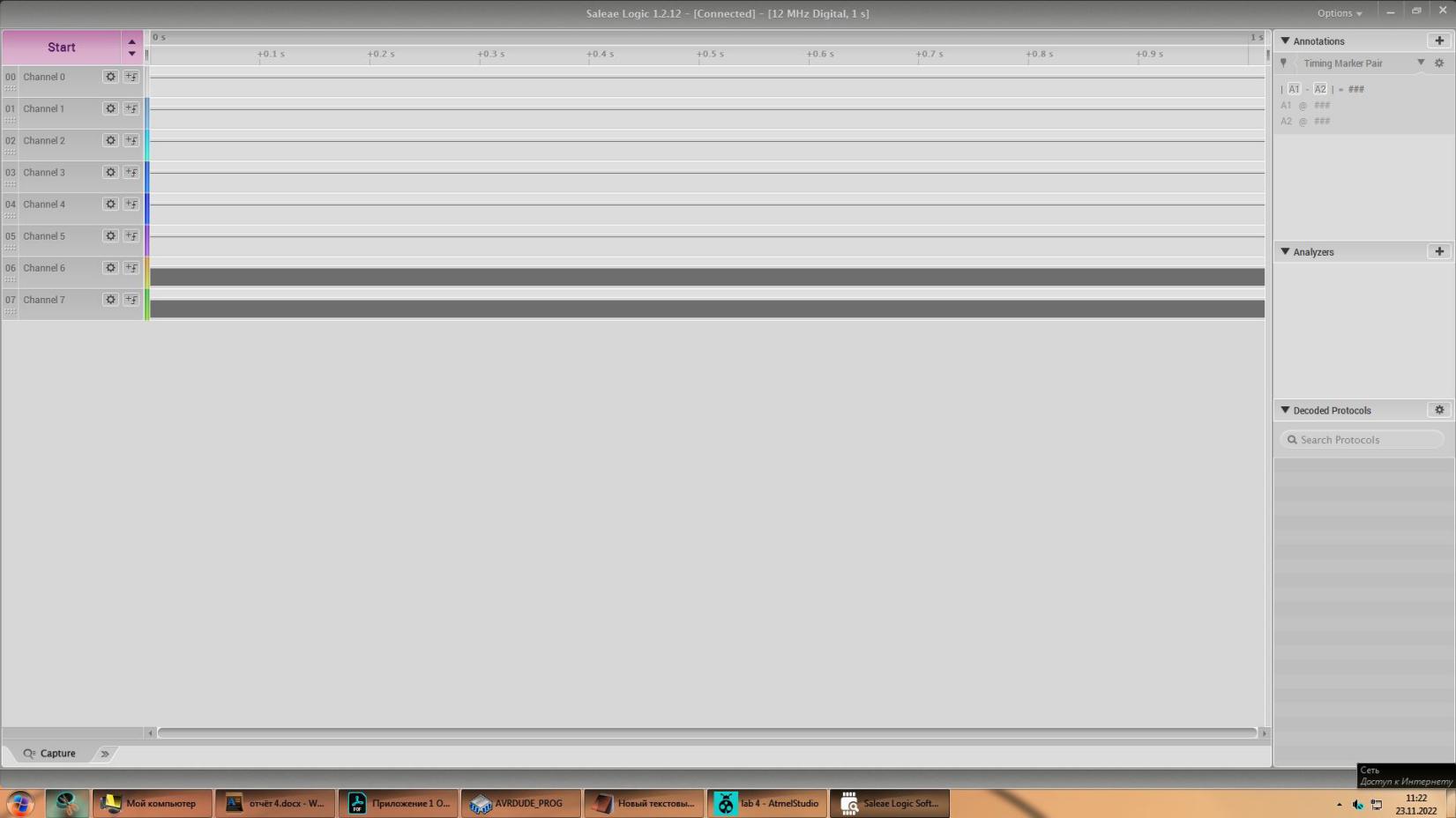


Рисунок 7 — выполнение задания 2 с потоковым выводом на Logic. 1

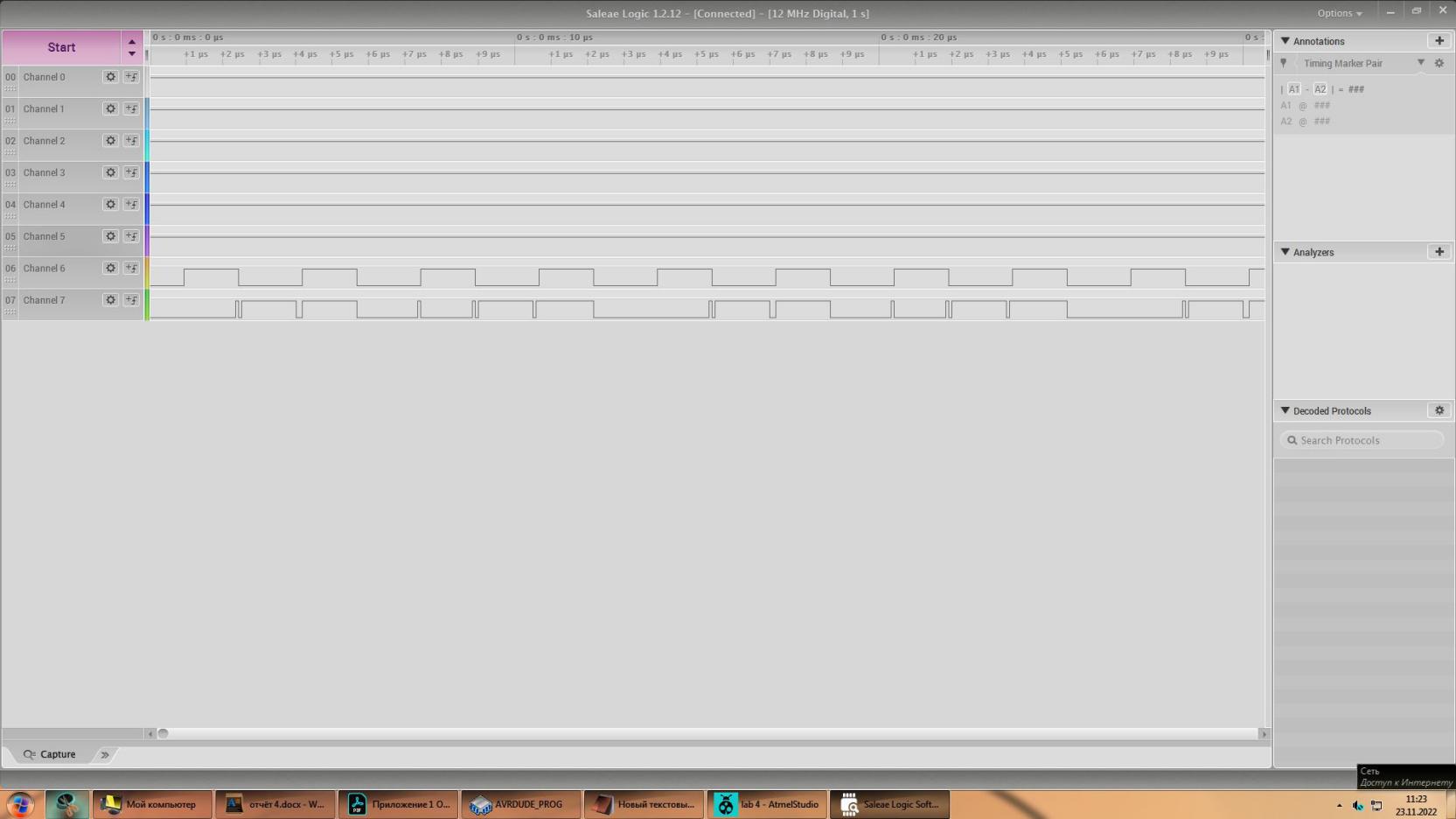


Рисунок 7 — выполнение задания 2 с потоковым выводом на Logic. 2