

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

по лабораторной работе № 2

Дисциплина: Микропроцессорные системы

Москва, 2022

Вариант 13.

Цель работы:

- изучение системы прерываний микроконтроллеров AVR,
- освоение системы команд микроконтроллеров AVR,
- ознакомление с работой стека при вызове подпрограмм и обработчиков прерываний,
- программирование внешних прерываний.

Ход работы

Задание 1.

Запустив AVR Studio, проверить работу программы в шаговом режиме. С целью ускорения отладки сократить время задержек до минимума. Проконтролировать работу стека при вызове подпрограмм delay1, delay2.

Убедившись в правильности работы программы восстановить параметры подпрограмм задержки и заново откомпилировать программу. Загрузить программу в память микроконтроллера и проверить её работу на плате.

Код программы:

```
;Соединения на плате STK500: SW0-PA0, SW1-PA1, LED0-PB0
;*****
.include "m8515def.inc" ;файл определений для ATmega8515
.def temp = r16 ;временный регистр
.equ led = 0 ;0-й бит порта PB
.equ sw0 = 0 ;0-й бит порта PA
.equ sw1 = 1 ;1-й бит порта PA

.org $000
    rjmp INIT ; обработка сброса
;***Инициализация МК***
INIT: ldi temp,$5F ; установка
      out SPL,temp ; указателя стека
      ldi temp,$02 ; на последнюю
      out SPH,temp ; ячейку ОЗУ
      ser temp ; инициализация выводов
      out DDRB,temp ; порта PB на вывод
      out PORTB,temp ; погасить LED
      clr temp ; инициализация
      out DDRA,temp ; порта PA на ввод
      ldi temp,0b00000011 ; включение 'подтягивающих'
      out PORTA,temp ; резисторов порта PA
```

```
test_sw0: sbic PINA,sw0 ;проверка состояния
          rjmp test_sw1 ; кнопки sw0
          cbi PORTB,led
          rcall delay1
          sbi PORTB,led
```

```
wait_0:  sbis PINA,sw0
          rjmp wait_0
```

```
test_sw1: sbic PINA,sw1 ;проверка состояния
          rjmp test_sw0 ; кнопки sw1
          cbi PORTB,led
          rcall delay2
          sbi PORTB,led
```

```
wait_1:  sbis PINA,sw1
          rjmp wait_1
          rjmp test_sw0
```

```
delay1:
;***Задержка (три вложенных цикла)***
ldi r17, 55
d1: ldi r18,95
   d2: ldi r19, 255
   d3: dec r19
   brne d3
   dec r18
   brne d2
   dec r17
   brne d1 ; подпрограмма 1 с
   ret
```

```
delay2:
;***Задержка (три вложенных цикла)***
ldi r17, 110
d11: ldi r18,95
   d12: ldi r19, 255
   d13: dec r19
   brne d13
   dec r18
   brne d12
   dec r17
   brne d11 ; подпрограмма 2 с
   ret
```

Расчёт задержки delay1:

$$T_{del1_1} = 1 + 255 \cdot (1+2) - 1 = 765 \text{ ЦИКЛОВ}$$

$$T_{del1_2} = 1 + 95 \cdot (765+1+2) - 1 = 72960 \text{ ЦИКЛОВ}$$

$$T_{del1_3} = 1 + 55 \cdot (72960 + 1 + 2) - 1 = 4012965 \text{ ЦИКЛОВ}$$

$$T_{delay1} = 4012965 : 4 \text{ МГц} = 4012965 \cdot 0,25 \text{ мкс} = 1003241,25 \text{ мкс} = \sim 1 \text{ с.}$$

На рисунках 1-2 показано количество пройденного времени до вхождения в подпрограмму delay1 и после выхода из неё.


A screenshot of a digital stop watch interface. It has a black background with white text. On the left, it says "Stop Watch" and on the right, it displays "4.25 us".

Рисунок 1 – время выполнения до входа в подпрограмму delay1

A screenshot of a digital stop watch interface. It has a black background with white text. On the left, it says "Stop Watch" and on the right, it displays "1003247.25 us".

Рисунок 2 – время выполнения после выхода из подпрограммы delay1

$$\text{Время задержки delay1} = 1003247.25 \text{ нс} - 4.25 \text{ нс} = \sim 1 \text{ с.}$$

Расчёт задержки delay2:

$$T_{del1_1} = 1 + 255 \cdot (1+2) - 1 = 765 \text{ ЦИКЛОВ}$$

$$T_{del1_2} = 1 + 95 \cdot (765+1+2) - 1 = 72960 \text{ ЦИКЛОВ}$$

$$T_{del1_3} = 1 + 110 \cdot (72960 + 1 + 2) - 1 = 8025930 \text{ ЦИКЛОВ}$$

$$T_{delay2} = 8025930 : 4 \text{ МГц} = 8025930 \cdot 0,25 \text{ мкс} = 2006482,5 \text{ мкс} = \sim 2 \text{ с.}$$

На рисунках 3-4 показано количество пройденного времени до вхождения в подпрограмму delay2 и после выхода из неё.

A screenshot of a digital stop watch interface. It has a black background with white text. On the left, it says "Stop Watch" and on the right, it displays "5.00 us".

Рисунок 3 – время выполнения до входа в подпрограмму delay2

A screenshot of a digital stop watch interface. It has a black background with white text. On the left, it says "Stop Watch" and on the right, it displays "2006489.25 us".

Рисунок 4 – время выполнения после выхода из подпрограммы delay2

$$\text{Время задержки delay2} = 2006489.25 \text{ нс} - 5.00 \text{ нс} = \sim 2 \text{ с.}$$

Схемы алгоритмов основной программы и обработки прерываний приведены на рисунке 5.

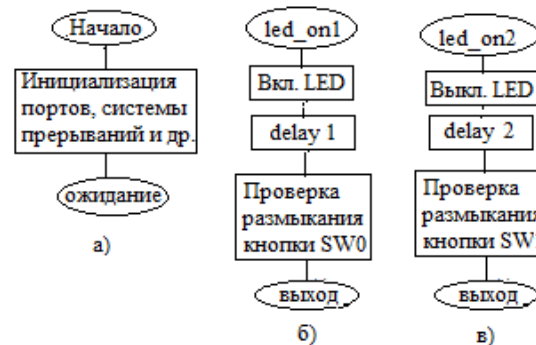


Рисунок 5 –Схемы алгоритмов основной программы (а) и прерываний (б, в)

Работа стека при прерываниях показана на рисунках 6-8.

The screenshot shows the AVR Studio IDE with the assembly code for the main program. The instruction pointer is at address 0x000000F, pointing to the `rcall delay1` instruction. The stack window on the right shows the current stack pointer at 0x1BA, with the stack containing several `FF` values. The status bar at the bottom indicates that there is no EEPROM data and that the file `C:\Users\fluch\Documents\AVR` is being deleted.

Рисунок 6 – Стек до передачи управления delay1 (rcall delay1)

The screenshot shows the AVR Studio IDE with the assembly code for the `delay1` routine. The instruction pointer is at address 0x000001B, pointing to the `ldi r17, 10:55` instruction. The stack window on the right shows the current stack pointer at 0x1BA, with the stack containing several `FF` values. The status bar at the bottom indicates that there is no EEPROM data and that the file `C:\Users\fluch\Documents\AVR` is being deleted.

Рисунок 7 – Стек после передачи управления delay1

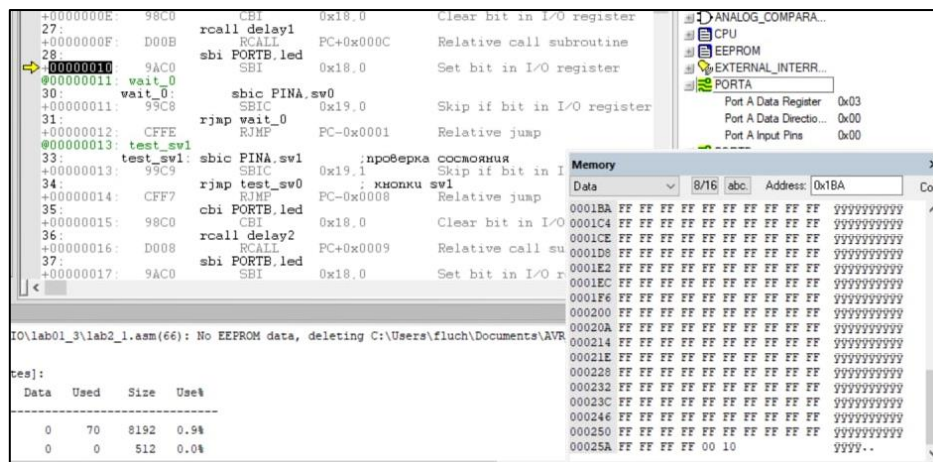


Рисунок 8 – Стек после возврата управления основной программе (ret)

Задание 2.

Для исследования механизма действия внешних запросов прерываний создадим проект, подключив кнопки SW0, SW1 к входам прерываний INT0, INT1 порта PD (PD2, PD3) микроконтроллера ATmega8515.

Вместо программной проверки состояний кнопок используем двухуровневую систему внешних прерываний микроконтроллера, на каждый уровень которой поступает запрос от кнопки. Схемы алгоритмов основной программы и обработки прерываний приведены на рисунке 9.

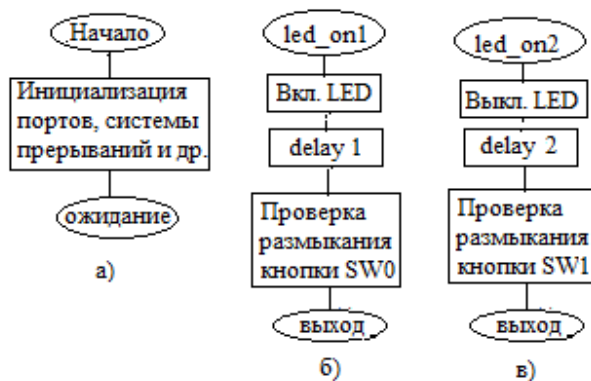


Рисунок 9 – Схемы алгоритмов основной программы (а) и прерываний (б, в)

Код программы:

```

;Соединения на плате STK500: SW0-PD2, SW1-PD3, LED0-PB0
;*****
.include "m8515def.inc" ;файл определений для ATmega8515
.def temp = r16 ;временный регистр
.equ led = 0 ;0-о бит порта PB
.equ sw0 = 2 ;2-й бит порта PD
.equ sw2 = 3 ;3-й бит порта PD
.org $000
  
```

```

;***Таблица векторов прерываний, начиная с адреса $000***
rjmp INIT      ;обработка сброса
rjmp led_on1   ;на обработку запроса INT0
rjmp led_on2   ;на обработку запроса INT1
;***Инициализация SP, портов, регистра маски***
INIT:
    ldi temp,$5F ;установка
    out SPL,temp ; указателя стека
    ldi temp,$02 ; на последнюю
    out SPH,temp ; ячейку ОЗУ
    ser temp ;инициализация выводов
    out DDRB,temp ; порта PB на вывод
    out PORTB,temp ;погасить СД
    clr temp ;инициализация
    out DDRD,temp ; порта PD на ввод
    ldi temp,0b00001100 ;включение 'подтягивающих'
    out PORTD,temp ; резисторов порта PD
    ldi temp,((1<<INT0)|(1<<INT1));разрешение прерываний
    out GICR,temp ; в 6,7 битах регистра маски GICR
    ldi temp,0 ;обработка прерываний
    out MCUCR,temp ; по низкому уровню
    sei ;глобальное разрешение прерываний
loop:
    nop ;режим ожиданий
    rjmp loop
led_on1:
    cbi PORTB,led
    rcall delay1
    sbi PORTB,led
wait_0:
    sbis pind,sw0
    rjmp wait_0
    reti
led_on2:
    cbi PORTB,led
    rcall delay2
    sbi PORTB,led
wait_1:
    sbis pind,sw2
    rjmp wait_1
    reti
delay1:
    ;***Задержка (три вложенных цикла)***
    ldi r17, 55
d1: ldi r18,95
    d2: ldi r19, 255
    d3: dec r19
    brne d3
    dec r18
    brne d2
    dec r17
    brne d1 ; подпрограмма 1 с
    ret

```

```

delay2:
;***Задержка (три вложенных цикла)***
ldi r17, 110
d11: ldi r18,95
    d12: ldi r19, 255
    d13: dec r19
    brne d13
    dec r18
    brne d12
dec r17
brne d11      ; подпрограмма 2 с
ret

```

Задержки работают аналогично задержкам в первой программе.

Работа стека при внешних прерываниях показана на рисунках 10-15.

До вызова внешнего прерывания мы находимся на адресе 13.

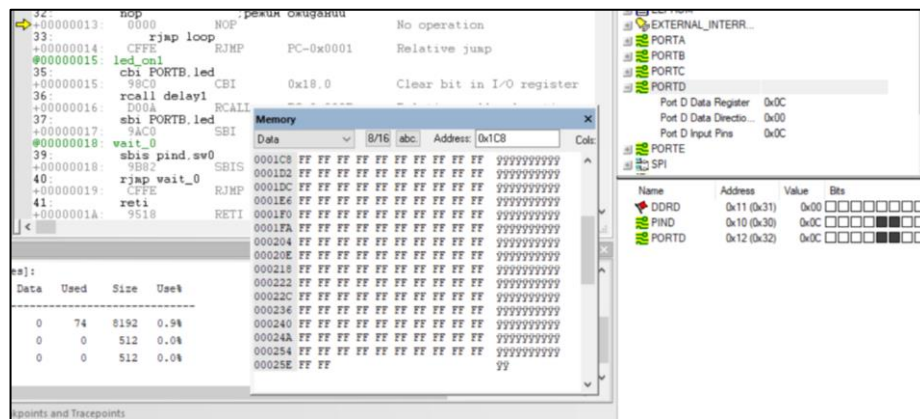


Рисунок 10 – Стек до вызова внешнего прерывания

При нажатии кнопки PIND2 (PIND3) происходит запросы внешнего прерывания INT0 (INT1), и мы переходим на адрес 1 (2). В стек заносится адрес 13, на котором мы были до вызова прерывания.

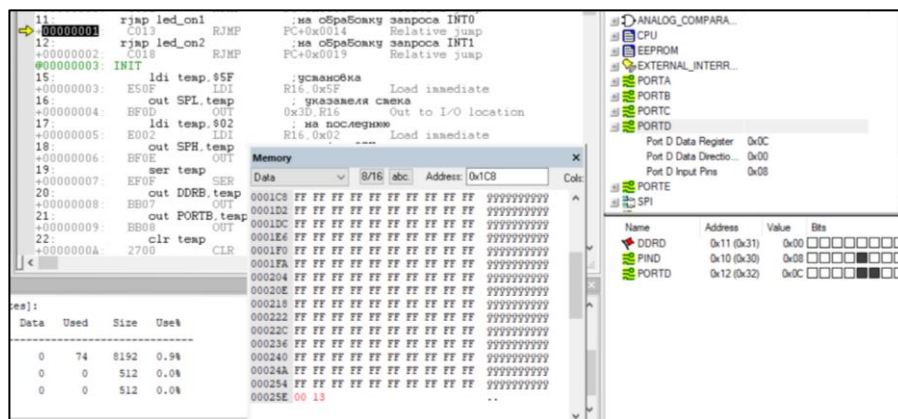


Рисунок 11 – Стек после вызова внешнего прерывания

Потом по безусловному преходу мы переходим на метку. Далее доходим до команды вызова подпрограммы.

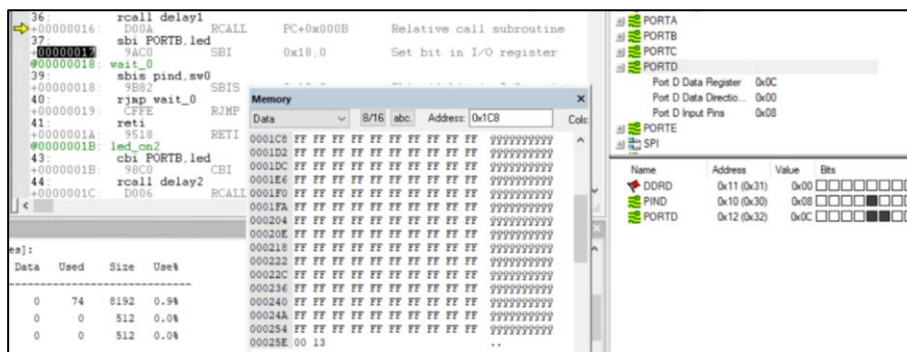


Рисунок 12 – Стек после вызова внешнего прерывания, до вызова подпрограммы

Происходит вызов подпрограммы и в стек записывается 17 - адрес возврата в основную программу.

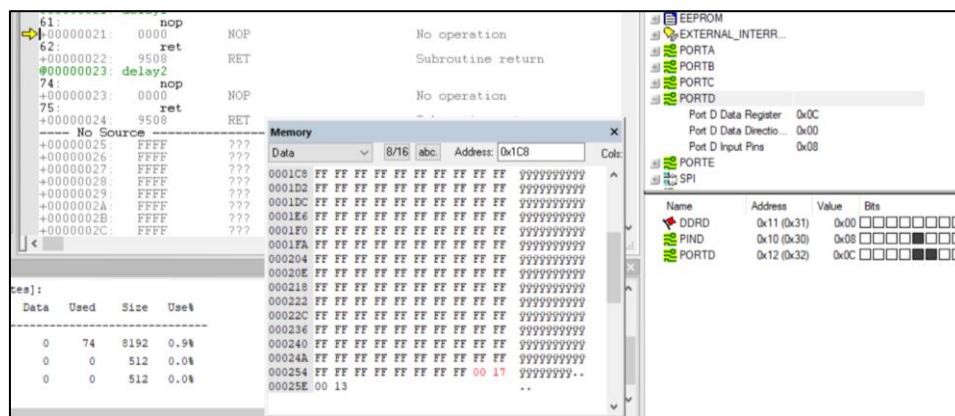


Рисунок 13 – Стек после вызова внешнего прерывания, после вызова подпрограммы

Затем по команде ret происходит возврат управления основной программе, переходим по адресу возврата, который лежит в вершине стека – адресу 17.

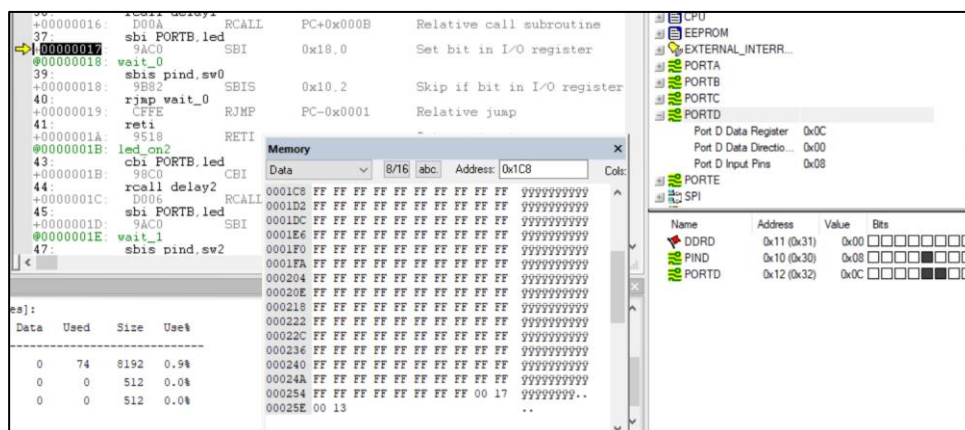


Рисунок 14 – Стек после вызова внешнего прерывания, после возврата управления

По команде `reti` происходит возврат из прерывания. Переходим на адрес 13 – адрес, лежащий на вершине стека.

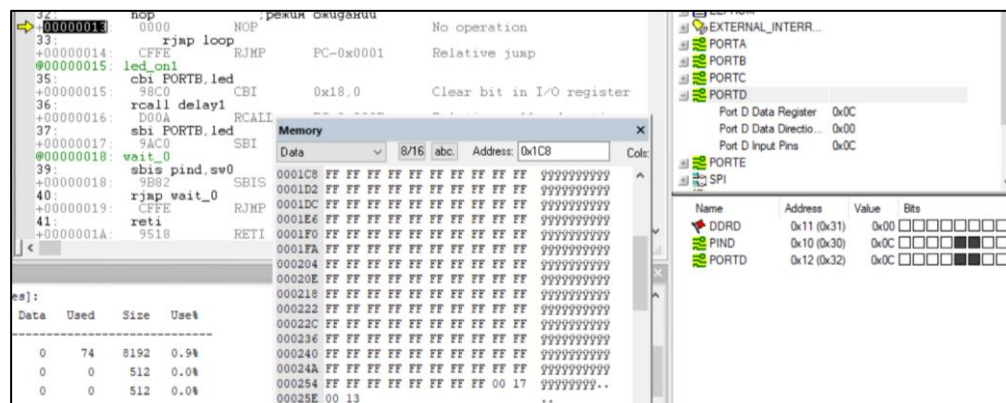


Рисунок 15 – Стек после возврата из внешнего прерывания

Задание 3.

Подготовим программу, соответствующую заданному алгоритму работы. При инициализации помимо общих директив устанавливаем исходный управляющий код в регистре индикации, нулевой разряд которого инициализирует зажигание светодиода, настраиваем на вывод порт микроконтроллера и указатель стека.

В цикле алгоритма на каждой итерации выполняем вывод в порт микроконтроллера управляющего слова, временную задержку, затем циклический сдвиг влево управляющего слова.

Код программы:

```
;Соединения на плате STK500: SW0-PD2, SW1-PD3, LED0-PA0
;*****
.include "m8515def.inc" ;файл определений для ATmega8515
.def job = r21 ;флаг выполнения
.def temp = r16 ;временный регистр
.def reg_led = r20 ;светодиоды
.equ sw0 = 2 ;2-й бит порта PD
.equ sw2 = 3 ;3-й бит порта PD
.org $000
;***Таблица векторов прерываний, начиная с адреса $000***
rjmp INIT ;обработка сброса
rjmp job_set ;на обработку запроса INT0
rjmp job_clr ;на обработку запроса INT1

;***Инициализация SP, портов, регистра маски***
INIT: ldi temp,$5F ;установка
out SPL,temp ;указателя стека
```

```

ldi temp,$02 ; на последнюю
out SPH,temp ; ячейку ОЗУ
ser temp ; инициализация выводов
out DDRA,temp ; порта PB на вывод
out PORTA,temp ; погасить СД
clr temp ; инициализация
out DDRD,temp ; порта PD на ввод
ldi temp,0b00001100 ; включение 'подтягивающих'
out PORTD,temp ; резисторов порта PD
ldi temp,((1<<INT0)|(1<<INT1)) ;разрешение прерываний
out GICR,temp ; в 6,7 битах регистра маски GICR
ldi temp,0 ;обработка прерываний
out MCUCR,temp ; по низкому уровню
ldi reg_led, 0x9F
ldi temp, 0xFF
ldi job, 0x00
sec
sei

```

```

loop:  sbrs job,0
       rjmp loop
       out PORTA,reg_led ;вывод на индикаторы
       rcall delay
MM:    brts LEFT ;переход, если флаг Т установлен
       sbrs reg_led,2 ;пропуск следующей команды,
       ;если 2-й разряд reg_led не установлен
       set ;T=1 - переключение флага направления
       ror reg_led ;сдвиг reg_led вправо на 1 разряд
       ;ror reg_led
       rjmp LOOP ;переход на проверку нажатия STOP

```

```

LEFT:  sbrs reg_led,5 ;пропуск следующей команды,
       ; если 5-й разряд reg_led не установлен
       clt ;T=0 – переключение флага направления
       rol reg_led ;сдвиг reg_led влево на 1 разряд
       ;rol reg_led
       rjmp LOOP
end:   rjmp loop

```

```

job_set: ldi job,1
wait_0: sbis pind,sw0
       rjmp wait_0
       reti

```

```

job_clr: clr job
wait_1: sbis pind,sw2
       rjmp wait_1
       reti

```

```

delay: ldi r17,11
d1:    ldi r18,95
d2:    ldi r19,255

```

```

d3: dec r19
brne d3
dec r18
brne d2
dec r17
brne d1
ret

```

Задание 4.

Составим программу согласно описанному алгоритму работы. Отладим работу программы в пошаговом режиме в среде AVR Studio.

Соберём схему для моделирования в среде ISIS Proteus.

Проверим работу программы, поочередно нажимая кнопки sw0 и sw1 и наблюдая состояние светодиода.

На рисунке 16 изображена схема собранная в среде ISIS Proteus.

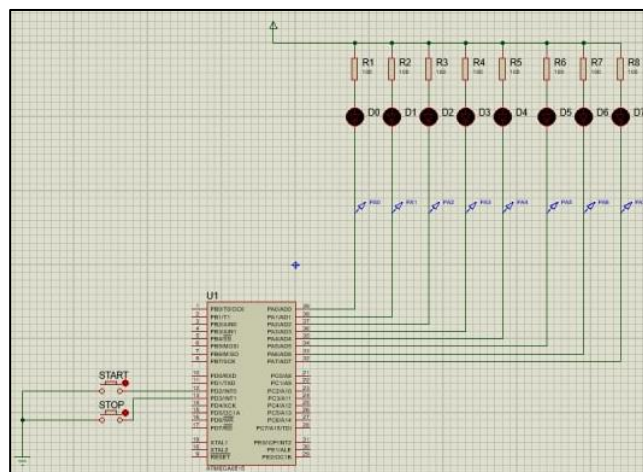


Рисунок 16 – Схема собранная в среде ISIS Proteus

Написанная нами программа работает корректно, это можно увидеть на диаграмме, изображенной на рисунке 17.

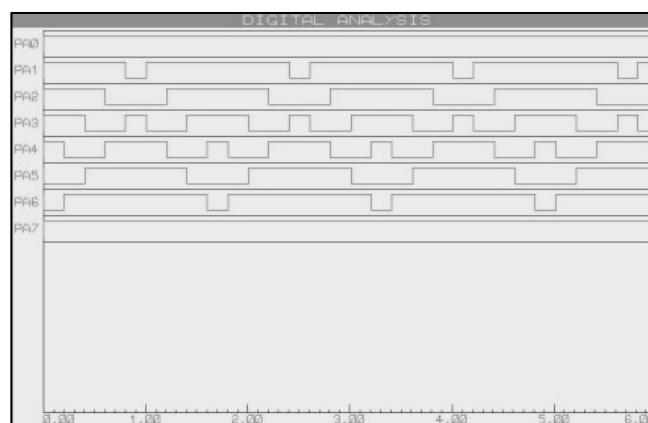


Рисунок 17 – Диаграмма работы программы

Задание 5.

Соберём схему на плате STK500. Загрузим программу в память микроконтроллера и проверить работу программы на макете.

Вывод

В результате выполнения лабораторной работы были изучены системы прерываний микроконтроллеров AVR, освоены системы команд микроконтроллеров AVR, был получен опыт работы со стеком при вызове подпрограмм и обработчиков прерываний и с программированием внешних прерываний. Также, были получены навыки работы со средой моделирования ISIS Proteus.