МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ АВТОМОБІЛЬНО-ДОРОЖНІЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з курсу
«Мікропроцесорні пристрої»
для студентів спеціальності 6.05020103
«Комп’ютерні системи управління рухомими об’єктами»

Харків
2015

Методичні вказівки до виконання лабораторних робіт з курсу «Мікропроцесорні пристрої» для студентів спеціальності 6.05020103 «Комп'ютерні системи управління рухомими об'єктами» / Сост.: О.С. Назаров. – Харків: ХНАДУ, 2015. – 76 с.

Укладач О.С. Назаров

Кафедра інформаційних технологій та мехатроніки

СОДЕРЖАНИЕ

ОБЩИЕ ПОЛОЖЕНИЯ

Целью изучения курса «Теория автоматического управления» является освоение студентами знаний по основам теории систем автоматического управления, состоящей из следующих основных частей: принципы построения, математическое моделирование и устойчивость линейных стационарных систем управления; теория нестационарных, цифровых и нелинейных систем автоматического управления.

В процессе выполнения лабораторных работ студенты осваивают методику исследования линейных и нелинейных, непрерывных и дискретных систем автоматического управления. При этом необходимо умение применять знания основных разделов курса «Теория автоматического управления».

Лабораторные работы выполняются студентами на персональных компьютерах. Студенты должны владеть основами алгоритмических языков или математических пакетов для ЭВМ.

В методических указаниях рассмотрены элементы и звенья систем автоматического регулирования, временные и частотные характеристики динамических звеньев, устойчивость линейных систем автоматического регулирования и др.

ТЕХНИКА БЕЗОПАСНОСТИ ПРИ РАБОТЕ С ЭВМ

1. К работе на ЭВМ допускаются студенты, изучившие правила техники безопасности, имеющие навыки работы с системой и текст программы для реализации.

2. Подготовка ЭВМ к работе, техническое обслуживание и ремонт производятся персоналом кафедры, имеющим соответствующую подготовку. Поэтому о любых неполадках в работе ЭВМ необходимо сообщить преподавателю.

3. Студентам при работе на ЭВМ разрешается пользоваться только клавиатурой, манипулятором «мышь» и дисплеем. Использование других устройств без разрешения преподавателя запрещается.

4. ***Категорически запрещается*** пользоваться дискетами, не проверенными преподавателем.

ЛАБОРАТОРНАЯ РАБОТА №1
ДВОИЧНАЯ СИСТЕМА СЧИСЛЕНИЯ.
ПРЕДСТАВЛЕНИЕ ДАННЫХ В ДВОИЧНОЙ СИСТЕМЕ.

Вопросы:

1. Позиционный двоичный код. Бит, тетрада, байт, слово.
2. Преобразование десятичных чисел в двоичные. Числа со знаком.
3. Шестнадцатеричные и двоично- десятичные числа.
4. Операции над двоичными кодами.
   - Арифметические операции (сложение, вычитание, инкремент, декремент). Признаки выполнения операций.
   - Логические операции (умножение, сложение, инверсия, сумма по модулю 2, сдвиг).
5. Кодирование информации с помощью двоичных кодов.

1. Позиционный двоичный код. Бит , тетрада, байт, слово.

Число в позиционной системе счисления с **n** разрядами

$$A=a_{n-1},a_{n-2}, \ldots,a_i, \ldots,a_1,a_0$$

имеет значение

$$A=a_{n-1}P^{n-1}+a_{n-2}P^{n-2}+, \ldots+a_i P^i+\ldots+a_1 P^1+a_0 P^0,$$

где: P- основание системы счисления;

$a_i=\{0, 1, \ldots P-2,P-1\}$ – значение разряда.

Десятичная система счисления (decimal – **Dec**).

$$[A]_{10}=a_{n-1}10^{n-1}+a_{n-2}10^{n-2}+, \ldots+a_i 10^i+\ldots+a_1 10^1+a_0 10^0.$$

Пример 1:

$216_{10}=2\times10^2+1\times10^1+6\times10^0 = 3\times100+1\times10+6\times1$ (n= 3).

Двоичная система счисления (binary – **Bin**)

$$[A]_2=a_{n-1}2^{n-1}+a_{n-2}2^{n-2}+, \ldots+a_i 2^i+\ldots+a_1 2^1+a_0 2^0.$$

$2^0=1,\ 2^1=2,\ 2^2=4,\ \ldots$

Задание 1: продолжить ряд до 2^{15}.

Пример2:

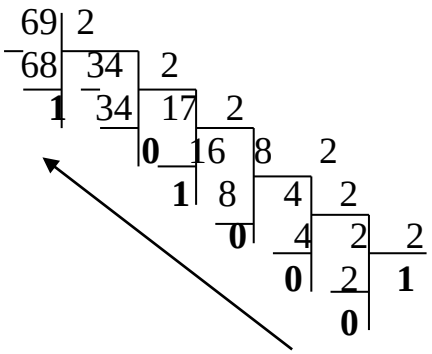
$11011000_2= 1\times2^7 + 1\times2^6+ 0\times2^5+ 1\times2^4+ 1\times2^3+ 0\times2^2+ 0\times2^1+ 0\times2^0=$
$=1\times128 + 1\times64+ 0\times32+ 1\times16+ 1\times8+ 0\times4+ 0\times2+ 0\times1= 216_{10}$ (n=8).

Задание 2. Получить значения двоичных чисел

1) 1111_2 (15_{10}, n=4 - тетрада)
2) 1000101_2 (69_{10}, n=7)
3) 01111111_2 (127_{10}, n=8 - байт)
4) 0000000010000000_2 (128_{10}, n=16 - слово)
5) 11111111_2 (255_{10}, n=8 - байт)

2. Преобразование десятичных чисел в двоичные. Числа со знаком.

Пример 3. Преобразование 69_{10} в 1010101_2 способом последовательного деления.



Задание 3. Выполнить преобразование десятичных чисел – 15, 127, 128, 255 в двоичные и сравнить результат с данными примера 2.

3. Шестнадцатеричные и двоично- десятичные числа.

Обозначение значений разрядов 16- ричных чисел:

0000 - 0	1000 - 8
0001 - 1	1001 - 9
0010 - 2	1010 - A
0011 - 3	1011 - B
0100 - 4	1100 - C
0101 - 5	1101 - D
0110 - 6	1110 - E
0111 - 7	1111 - F

Пример 4. Преобразование двоичного (Bin) кода 000110101_2 в шестнадцатеричный (Hex) код 035_{16} и обратно.

$000110101_2 \rightleftarrows (0000)(0011)(0101) \rightleftarrows 035_{16}$= \$035 = 035h= 0x035

Задание 4. Выполнить преобразование Bin – Hex для кодов

1) $0011000110111010011 1_2$ ($31BA7_{16}$= 31BA7h=203687)
2) 01101001
3) 11111111
4) 0000110011101101

Пример 5. Преобразование десятичного числа 3309 в двоично- десятичное число и обратно.

$3309_{10} \rightleftarrows (3)(3)(0)(9) \rightleftarrows (0011)(0011)(0000)(1001) \rightleftarrows 0011001100001001_{2\text{-}10}$

Задание 5. Выполнить преобразование для кодов:

1) 184317
2) 234
3) 795724
4) 1234567

Операции над двоичными кодами.

1) Арифметические операции.
   - Инкремент (Inc) (+) $A + 1 \rightarrow A$

   Сложение двоичных разрядов выполняется по правилу:

   0 + 0 + 0 = 0, перенос 0
   0 + 0 + 1 = 1, перенос 0
   0 + 1 + 0 = 1, перенос 0
   0 + 1 + 1 = 0, перенос 1
   1 + 0 + 0 = 0, перенос 0
   1 + 0 + 1 = 1, перенос 1
   1 + 1 + 0 = 1, перенос 1
   1 + 1 + 1 = 0, перенос 1

Здесь определены значения суммы и значение переноса в следующий разряд.

Пример 6:

Прибавим к числу 69 (01000101) единицу.

0000001 значение разряда переноса
+ 01000101 (69_{10})
  00000001 (1)
  01000110 (70_{10})
  | | \
  64 4 2 $\rightarrow$ 70_{10}

Прибавим к числу A = 127_{10} (01111111_2) единицу.

1111111 значение разряда переноса
+ 01111111 (127_{10})
  00000001
  10000000 $\rightarrow$ 128_{10}

Прибавим к числу A = 255_{10} (11111111_2) единицу.

+ 11111111 (255_{10})
  1
  100000000
  |

C – признак переполнения 8-разрядной сетки

При переполнении разрядной сетки результат будет неверный, если не учитывать значение переноса.

Задание 6:
Сложить два числа 47_{10} и 25_{10}.
Сложить два числа 47_{10} и 220_{10}.
2) Операция вычитания
- Декремент (Dec) A - 1 → A

Пример 7:
Вычесть из числа 68_{10} (01000100_2) единицу.

```
  0000010   - разряд заема
_ 0100100
  0000001
  0100011
```

Задание 7:
Вычесть из числа 172_{10} (10101100_2) единицу.
Вычесть из числа 107_{10} (01101011_2) единицу.

Логические операции

Логические операции над двоичными кодами являются поразрядными, т.е. значение результата вычисления разряда не зависит от других разрядов.

Пример 8:

Выполнить логическое умножение, сложение, сложение по модулю два чисел 47_{10} (0010.1111_2) и 220_{10} (1101.1100_2) и операцию **НЕ** над этими числами.

	1	1	0	0	разряд кода 1
	1	0	1	0	разряд кода 2
или	1	1	1	0	результат поразрядного логического сложения
и	1	0	0	0	результат поразрядного логического умножения
⊕	0	1	1	0	результат поразрядного сложения по модулю 2

ИЛИ	И	НЕ	
0010.1111	0010.1111	0010.1111	1101.1100
1101.1100	1101.1100	1101.0000	0010.0011
1111.1111	0000.1100		

Задание 8:

Выполнить логическое умножение, сложение, сложение по модулю два чисел 231_{10} (1110.0111_2) и 87_{10} (0101.0111_2) и операцию **НЕ** над этими числами.

Выделение значения одного бита из байта.

Для выделение одного бита подбирается маска. В результате логического умножения все разряды кроме выделенного обнуляются. Если результат равен нулю, то выделенный бит нулевой, если результат не равен нулю, то бит единичный.

Пример 9:
Выделить бит 5 кода 0010.1111. Необходима маска 0010.0000. В результате логического умножения маски и кода получим:

0010.1111
0010.0000
0010.0000 - результат не равен 0, значит бит 5 кода равен 1.

Выделить бит 1 кода 1101.1100. Необходима маска 0000.0010. В результате поразрядного логического умножения маски и кода получим:

1101.1100
0000.0010
0000.0000 - результат равен 0, значит бит 1 кода равен 0.

Задание 9:
Выделить бит 4 кода 11011010_2 (218_{10})

Выборочная очистка бита.

Пример 10:

Для очистки бита 3 кода 01011101_2 (93_{10}) необходимо выполнить операцию **И** с маской **1111.0111**. В результате получим

0101.1101
1111.0111
0101.0101

Задание 10: Очистить бит 5 кода 01001101 и бит 1 кода 01110001

Выборочная установка бита.

Пример 11:

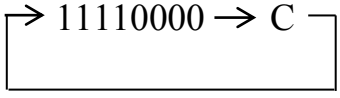
Для установки бита 3 при сохранении без изменения все остальных разрядов необходимо выполнить операцию **ИЛИ** с маской . В такой маске должны быть все нули кроме выделенного разряда - **00001000.**

ИЛИ
10111010
00001000 — Маска
10111010

Задание 11:
Установить бит 6 кода 00101001 сохранив без изменения все остальные разряды.
Выборочное инвертирование бита.
Пример 12:
Изменить состояние одного бита на противоположное, сохранив остальные.
Это можно сделать, если к коду прибавить по модулю два маску с единицей в выделенном разряде. (00001000)
10111010
00001000 — Маска
10110010 — Результат
00001000 — Маска
10111010
Задание 12:
Изменить состояние бита 4 кода 01001111 на противоположное, сохранив остальные.
Операции сдвига.
Сдвиги → Логические
↘ Циклические
Логический сдвиг влево.

Пример 13:
C ←
0 ← 00001111 ← 0
0 00011110
0 00111100
0 01111000
0 11110000
1 11100000
1 11000000
1 10000000
1 00000000

Логический сдвиг вправо.
→
0 → 11110000 → C
Циклический сдвиг.



Задание 13:
Выполнить 8 последовательных логических и циклических сдвигов байта $55 = 55h.

ЛАБОРАТОРНАЯ РАБОТА №2
РАЗРАБОТКА И ОТЛАДКА ПРОГРАММ ДЛЯ AVR-МИКРОКОНТРОЛЛЕРОВ В СРЕДЕ ГРАФИЧЕСКОГО ПРОГРАММИРОВАНИЯ ALGORITHM BUILDER

Программа работы:
1. Изучение элементов среды Algorithm Builder
2. Изучение шаблонов операций
3. Ввод и исследование учебной программы средствами Algorithm Builder
4. Разработка и исследование проектов программ обработки данных
5. Разработка и исследование проекта программы преобразования формы представления данных (двоичной в двоично-десятичную)

1. Изучение элементов среды Algorithm Builder.

По описанию среды ознакомиться с элементами алгоритма :

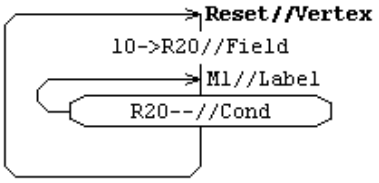


Рисунок1. Элементы среды (пример)

VERTEX – Вершина блока;
FIELD – Поле;
LABEL – Метка;
CONDITION – Условный переход;
JMP Vector – Относительный безусловный переход;
TEXT – Строка локального текстового редактора.

Для каждого указанного элемента и комментариев выделить способы размещения его в графическое поле построения программы.

2. Изучение шаблонов операций

Ознакомиться с описанием шаблонов операций, перечисленным в таблице 1. Команды разделены на группы операций - с одним регистром **R** (**R** 0 .. **R** 31); двумя регистрами **R - R** ; регистровыми парами **R R** (**R 24: R 25 – W**, **R 26: R 27 – X, R 28: R 29 – Y, R 30: R 31 – Z)**, как 16- разрядными регистрами; битами флагов и портов; условных переходов; пропусков; специальные. Для каждого шаблона из [1,с. 9-11] сделать выборку – действие, примеры и аналоги – команды AVR микроконтроллера, мнемоническое обозначение которых выделено курсивом (*LDI R,K, . MOV R,R* и др.), например:

Загрузка и пересылка данных

1) **# -> R** Загрузка константы # в рабочий регистр
24 -> r16 *LDI R,K*

2) **R -> R** Копирование одного рабочего регистра в другой
R0 -> R1 *MOV R,R и т.д.*

Таблица 1. Список шаблонов операций

Регистр	Регистр-регистр	Пара регистров	Бит
# -> R	R -> R	# -> RR	0 -> C
R + #	R + R	RR - #	1 -> C
R - #	R – R	RR + #	1 -> P.#
R++	R -> P	RR -> RR	0 -> P.#
R--	P -> R		
R & #	R & R	Условия	перехода
R ! #	R ! R	=	-=
|R	R ^ R	C = 0	C = 1
R &| #		>=	<
<<R		Условия	пропуска
<<R<	Специальные	R = R	
R>>	NOP	R.# = 0	R.# = 1
>R>>	RET	P.# = 0	P.# = 1

3. Ввод и исследование учебной программы средствами Algorithm Builder

Создать в D:\МП ПСА\ папку work для размещения разрабатываемых проектов программ. Целесообразно каждый проект Prog1, Prog2 … размещать в собственной папке D:\МП ПСА\ work\Prog1\, D:\МП ПСА\ work\Prog2\, …

Ознакомиться с порядком создания и отладки программы [1, с. 6-9]. Самостоятельно запустить **Algorithm Builder (ПУСК – Все программы - Algorithm Builder – v511 - Algorithm Builder**) и воспроизвести учебный пример как Prog1. Разрешить создание из исходного представления программы на Assembler (рис.2).

Необходимые элементы программной модели - **Processor, Working Registers, Port B, Process Time #0** открываются из списка в пункте меню **Открыть** (View).

При выполнении программы оценить время между точками останова при тактовой частоте 4 Мгц.

В отчет поместить исследуемую программу и оценки ее временных параметров. В рабочей папке найти текстовые файлы Prog1.hex, Prog1.asm и поместить их в отчет.

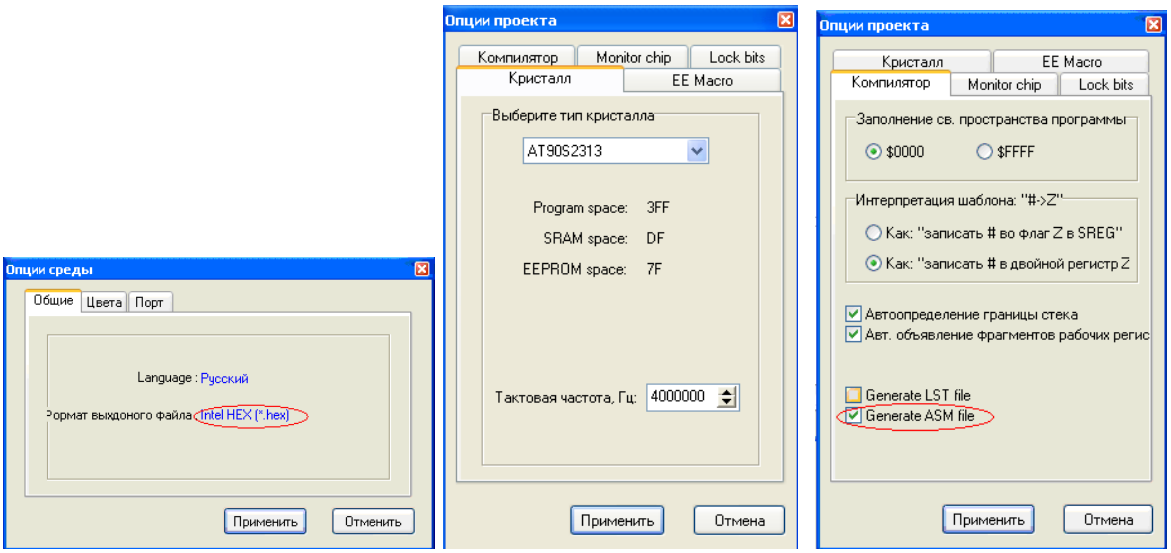
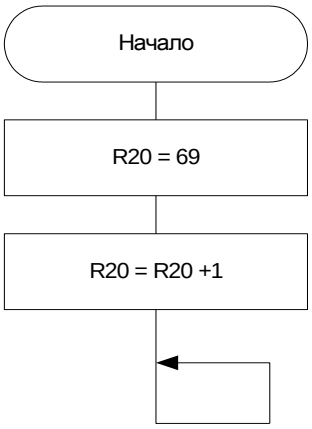


Рисунок 2. Опции среды и проекта для создания HEX- файла и ASM- файла

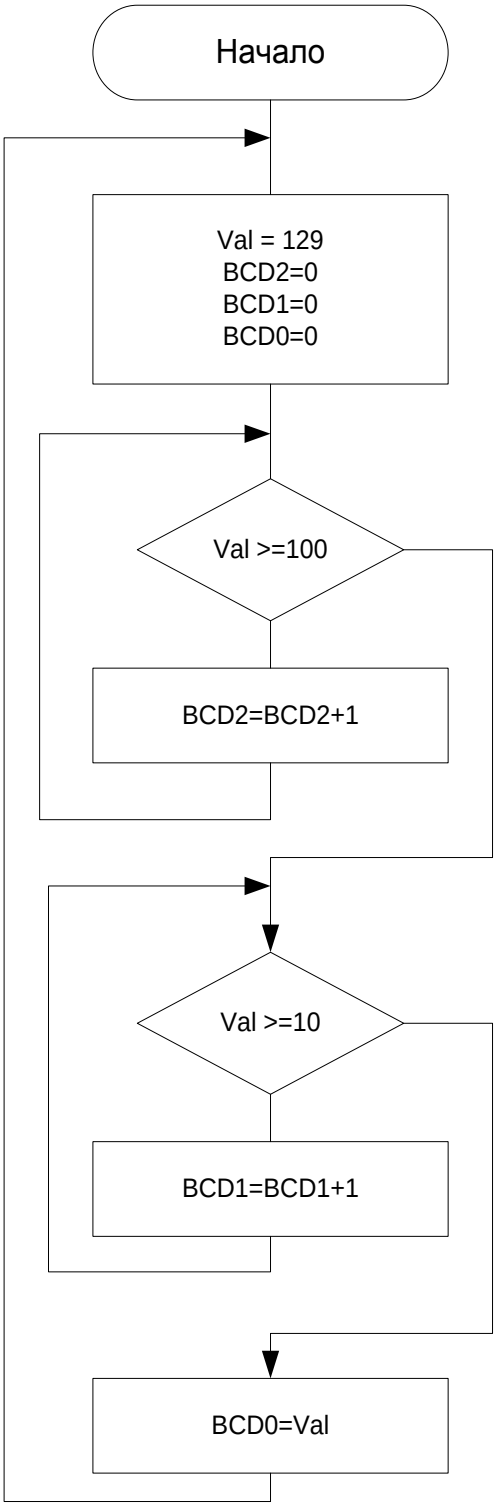
4. Разработка и исследование проектов программ обработки данных

По приведенной схеме алгоритма построить и отладить программу для выполнения арифметической операции из примера 6 занятия 1. Построить программы для выполнения операций из примеров 7-11 занятия 1 и сравнить результаты в различных системах счисления.



5. Разработка и исследование проекта программы преобразования формы представления данных (двоичной в двоично-десятичную)

По приведенной схеме алгоритма построить и отладить программу преобразования значения Val=129 в двоично- десятичные числа BCD2=1, BCD1=2, BCD0=9. Имена Val, BCD2, BCD1, BCD0 присвоить рабочим регистрам. Проверить работу программы для других значений.



Содержание отчета

- Шаблоны операций и условий перехода
- Фрагменты исследуемых алгоритмов
- Выводы по работе

ЛАБОРАТОРНАЯ РАБОТА №3
ПРОГРАММИРОВАНИЕ ОПЕРАЦИЙ ВВОДА-ВЫВОДА

Цель работы: Приобретение практических навыков программирования и отладки операций вывода циклических последовательностей комбинаций сигналов и управление выводом в соответствии с входными сигналами. Загрузка программы в память микроконтроллера стенда и проверка работы программы.

Состав стенда:

- Объект управления – шаговый двигатель ПБМГ-200-265
- Источники входных сигналов – замыкающие кнопки
- Микроконтроллер – AT90S2313 (ATtiny2313).

Среда программирования и отладки – Algorithm Builder v.5.11.

Общие сведения

Схема управления четырехфазным униполярным шаговым двигателем

Схема на рис. 1 упрощена. Отсутствуют силовые ключи, кварцевый резонатор и цепи питания МК.

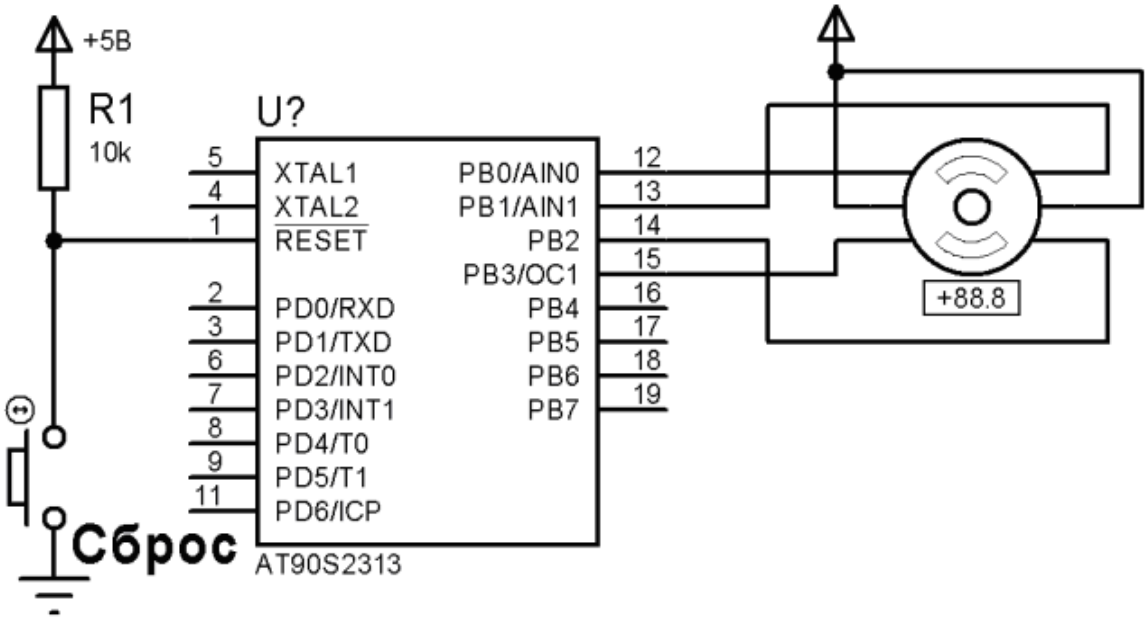


Рисунок 1. Упрощенная схема управления
четырехфазным униполярным шаговым двигателем

R1 - подтягивающий резистор 5 .. 50 Ком (Обязателен!!!)

RESET = 1 - ШД вращается (кнопка отпущена)

RESET = 0 - ШД останавливается (кнопка нажата, МК не работает)

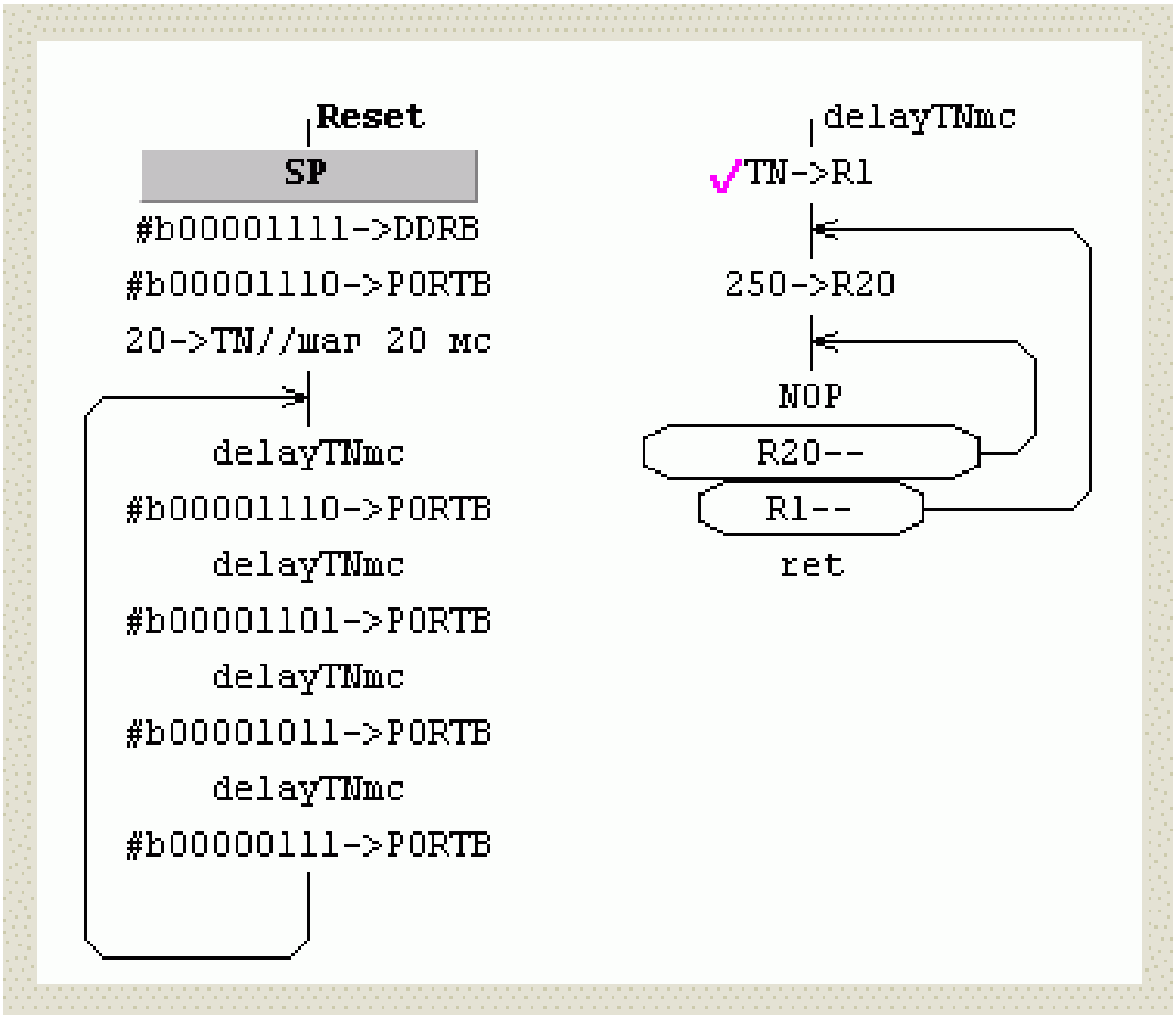
.Распределение выводов МК :

- F1 - PB0
- F2 - PB1
- F3 - PB2
- F4 - PB3

Таблица 1. Чередование фаз

F4	F3	F2	F1
1	1	1	0
1	1	0	1
1	0	1	1
0	1	1	1
1	1	1	0

Программа для формирования последовательности комбинаций сигналов F1-F2-F3-F4-F1- … приведена ниже. В каждом цикле выполняется 4 шага.



Стенд для исследования управления ШД

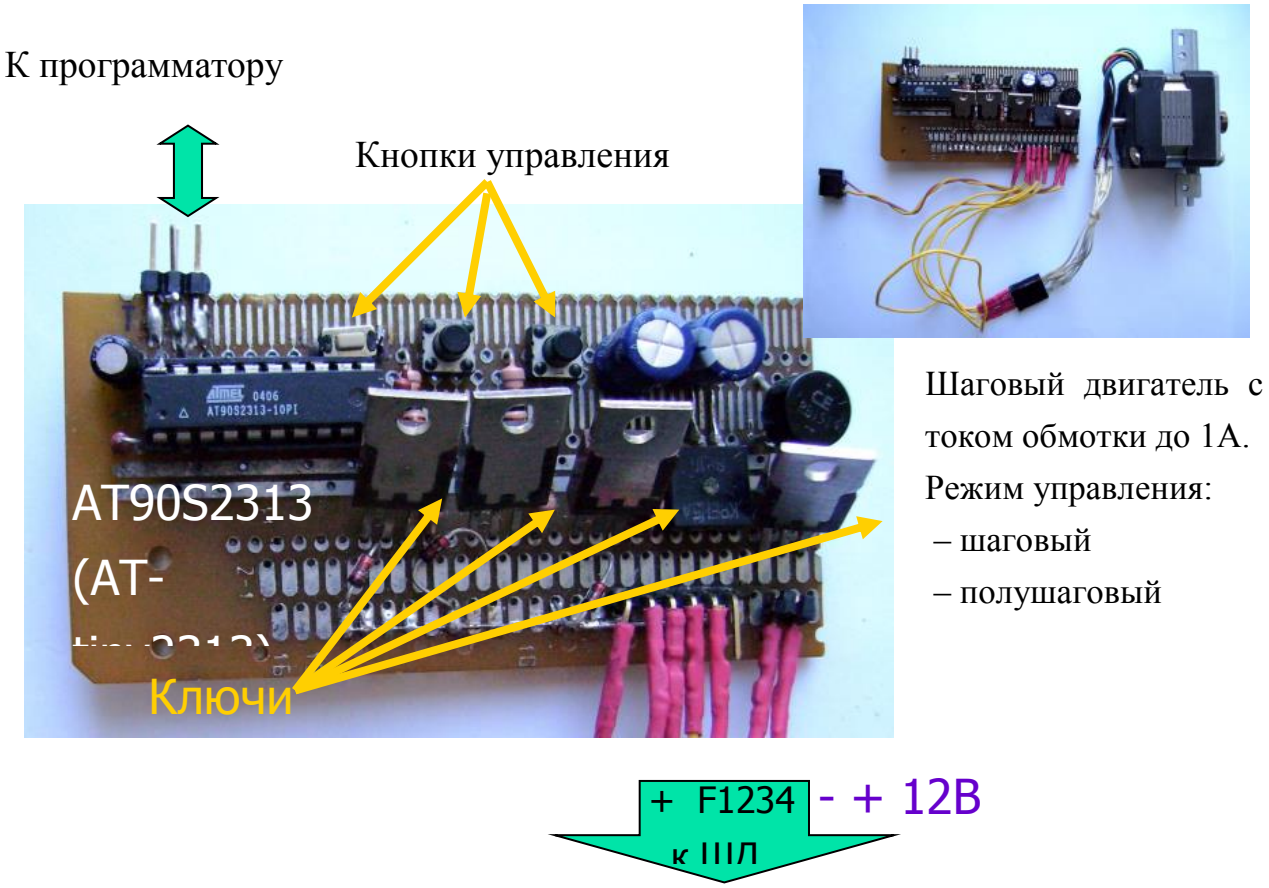


Рисунок 2. Стенд для исследования программ управления ШД

Модель простейшей управления ШД от AVR МК в среде PROTEUS

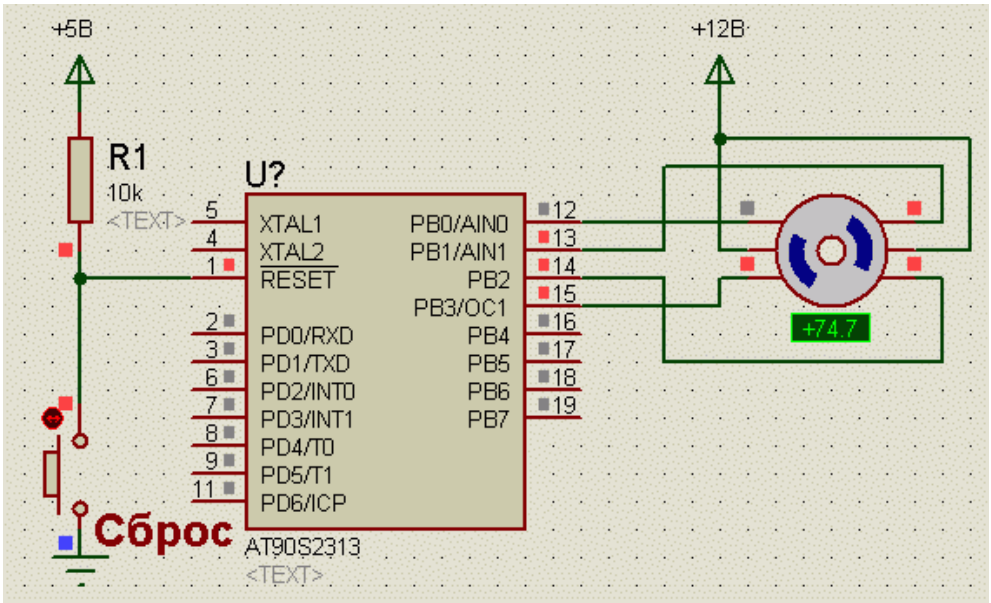


Рисунок 3. Модель для исследования управления ШД

Программа 1 с циклом формирования одиночного шага

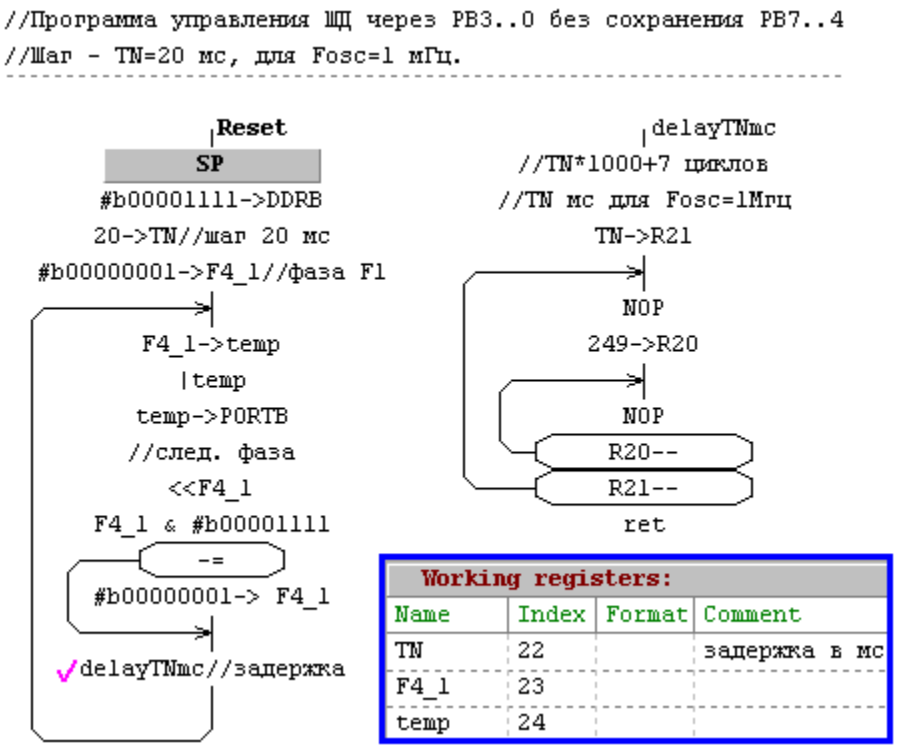


Рисунок 4. Простейшая программа управления ШД в полношаговом режиме

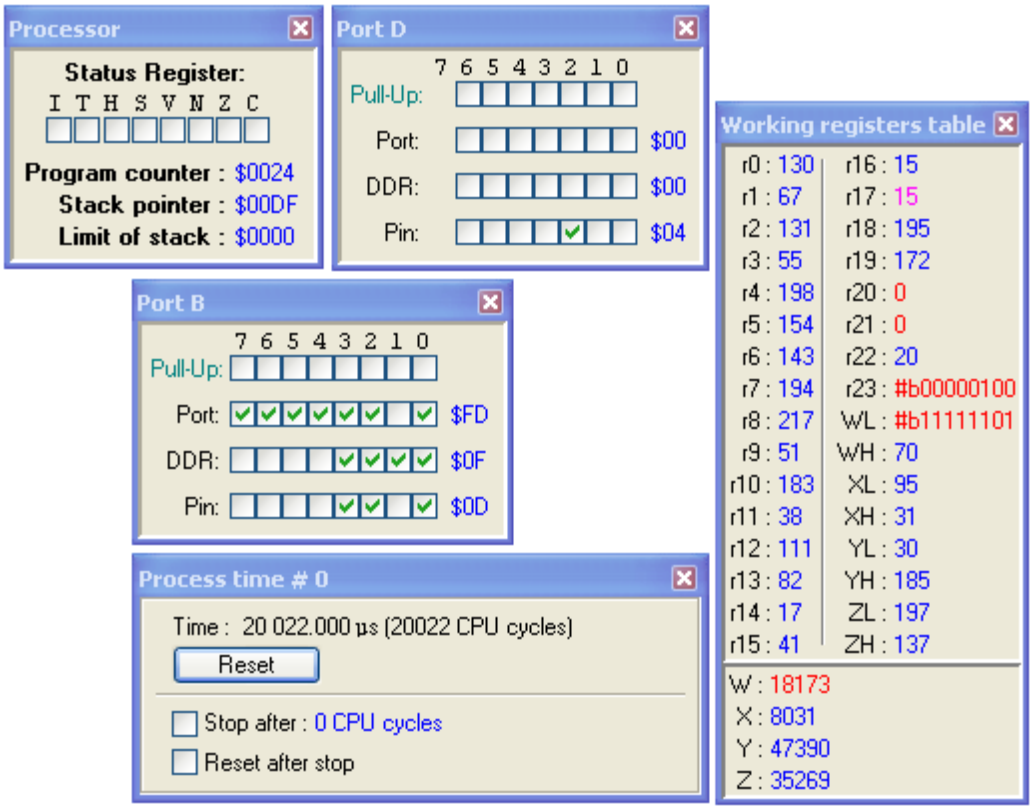


Рисунок 5. Программная модель в симуляторе при отладке программы

Программа 2 с реализацией цикла «чтение-модификация-запись»

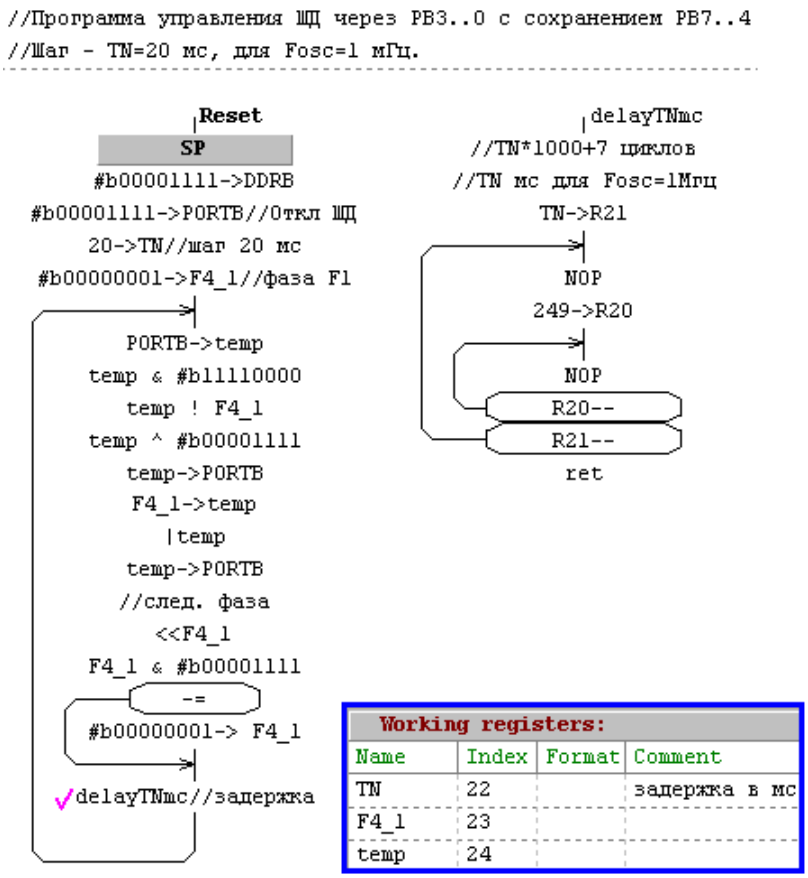


Рисунок 6. Усовершенствованная программа управления ШД

Простейший интерфейс оператора

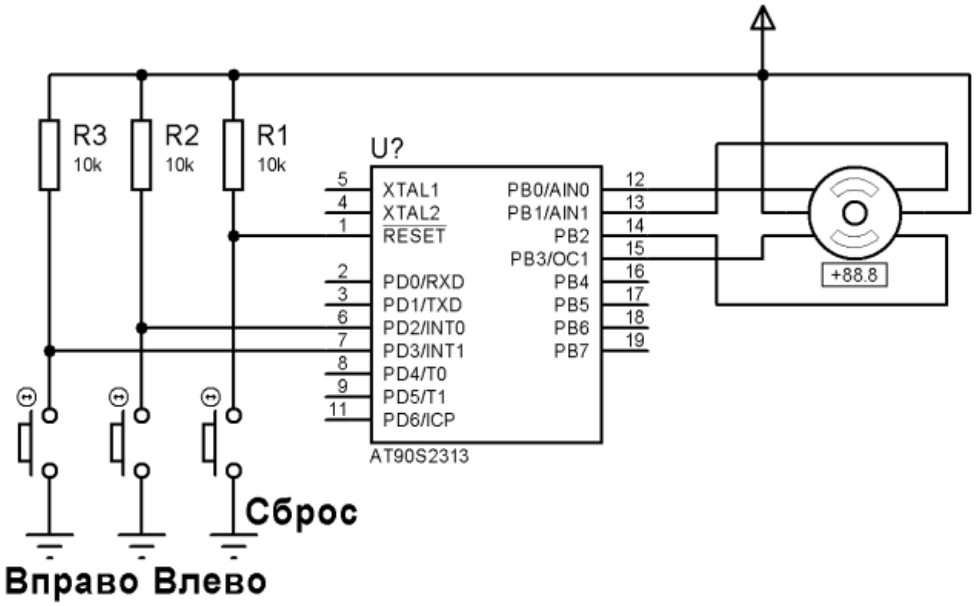


Рисунок 6. Простейший интерфейс оператора

Модель кнопочного управления ШД от AVR МК в среде PROTEUS

Включение питания, **Сброс** – останов ШД, выбор направления вращения нажатием кнопок.

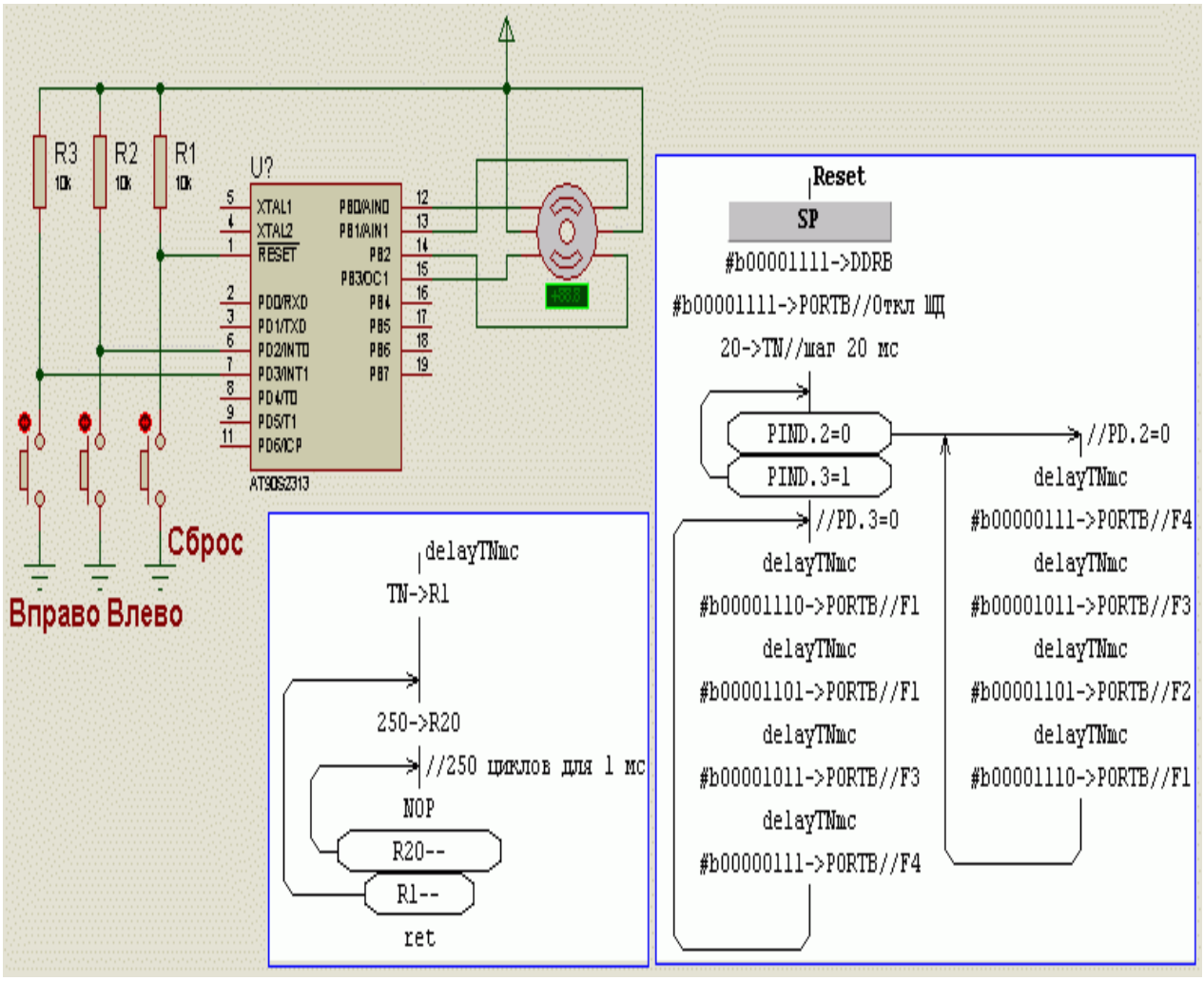
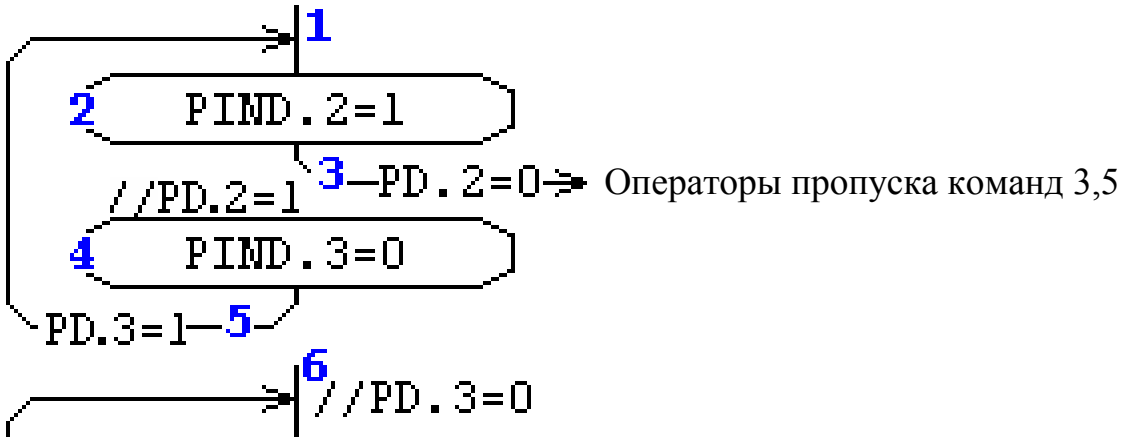


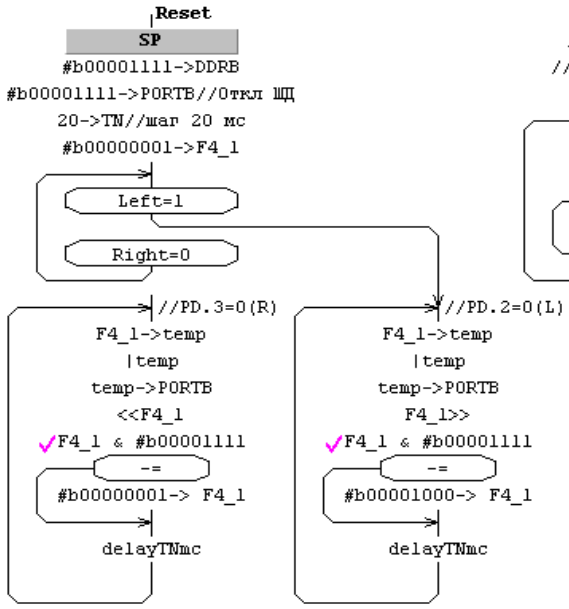
Рисунок 7. Модель управления ШД от AVR МК в среде PROTEUS



Операторы пропуска команд 3,5

Программа 3 управления ШД с выбором направления вращения

```
//Программа управления ШД с выбором направления вращения
//PD.2=0(Left=0) - влево, PD.3=0(Right=0) - вправо, Reset - останов. Шаг - TN=20 мс, для Fosc=1 мГц.
//TN, F4_1, temp - имена рабочих регистров (объявляются в секции Working registers)
//Имена линий кнопок Left (PD.2) и Right (PD.3) объявляются в секции Bits
```



Working registers:

Name	Index	Format	Comment
TN	22		задержка в мс
F4_1	23		
temp	24		

EEPROM:

Name	Address	Format	Count	Value

I/O registers:

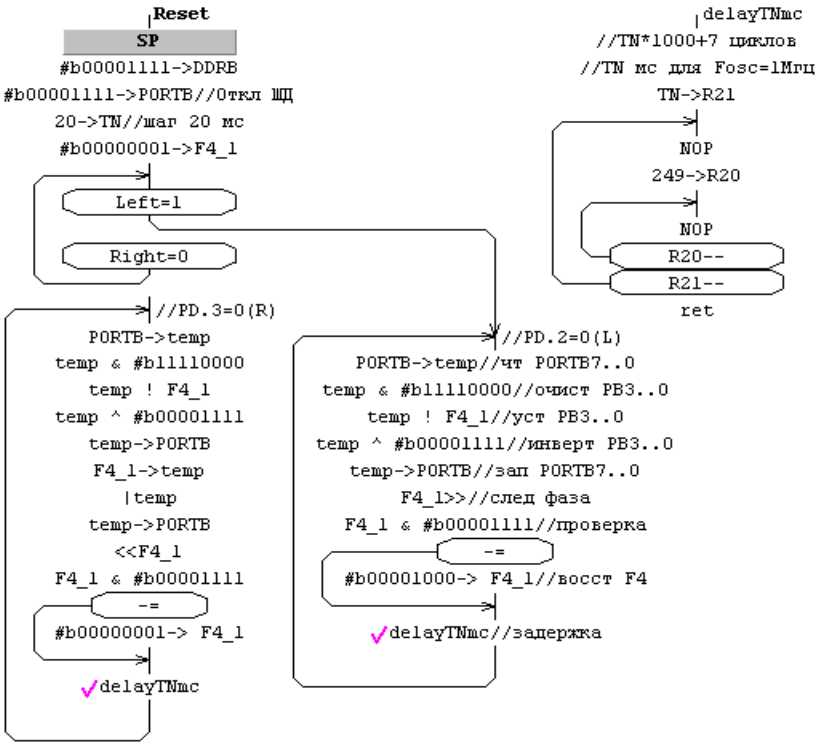
Name	I/O regist	Comment

Bits:

Name	Bit	Comment
Left	PIND.2	
Right	PIND.3	

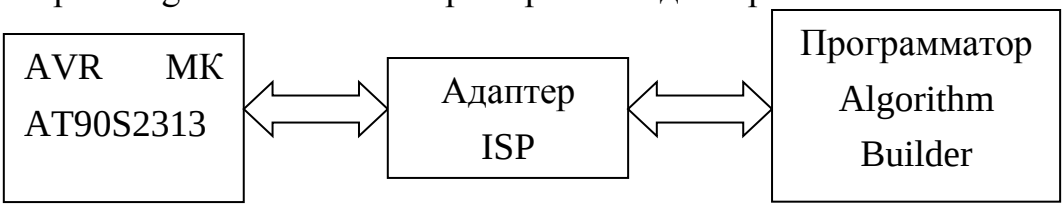
Программа 4 управления ШД с выбором направления вращения

```
//Программа управления ШД через PB3..0 с выбором направления вращения
//PD.2=0(Left=0) - влево, PD.3=0(Right=0) - вправо, Reset - останов. Шаг - TN=20 мс, для Fosc=1 мГц.
//TN, F4_1, temp - имена рабочих регистров (объявляются в секции Working registers)
//Имена линий кнопок Left (PD.2) и Right (PD.3) объявляются в секции Bits
```



Внутрисхемное программирование

Загрузка программы в память микроконтроллера выполняется программатором Algorithm Builder через простой адаптер



Для подключения адаптера в схеме микроконтроллерного устройства предусматривается специальный разъем. На него выводится сигнал сброса RST и 3 информационных сигнала последовательного интерфейса внутрисхемного программирования (ISP) – MOSI, MISO, SCK.

Порядок программирования МК изложен в описании Algorithm Builder.

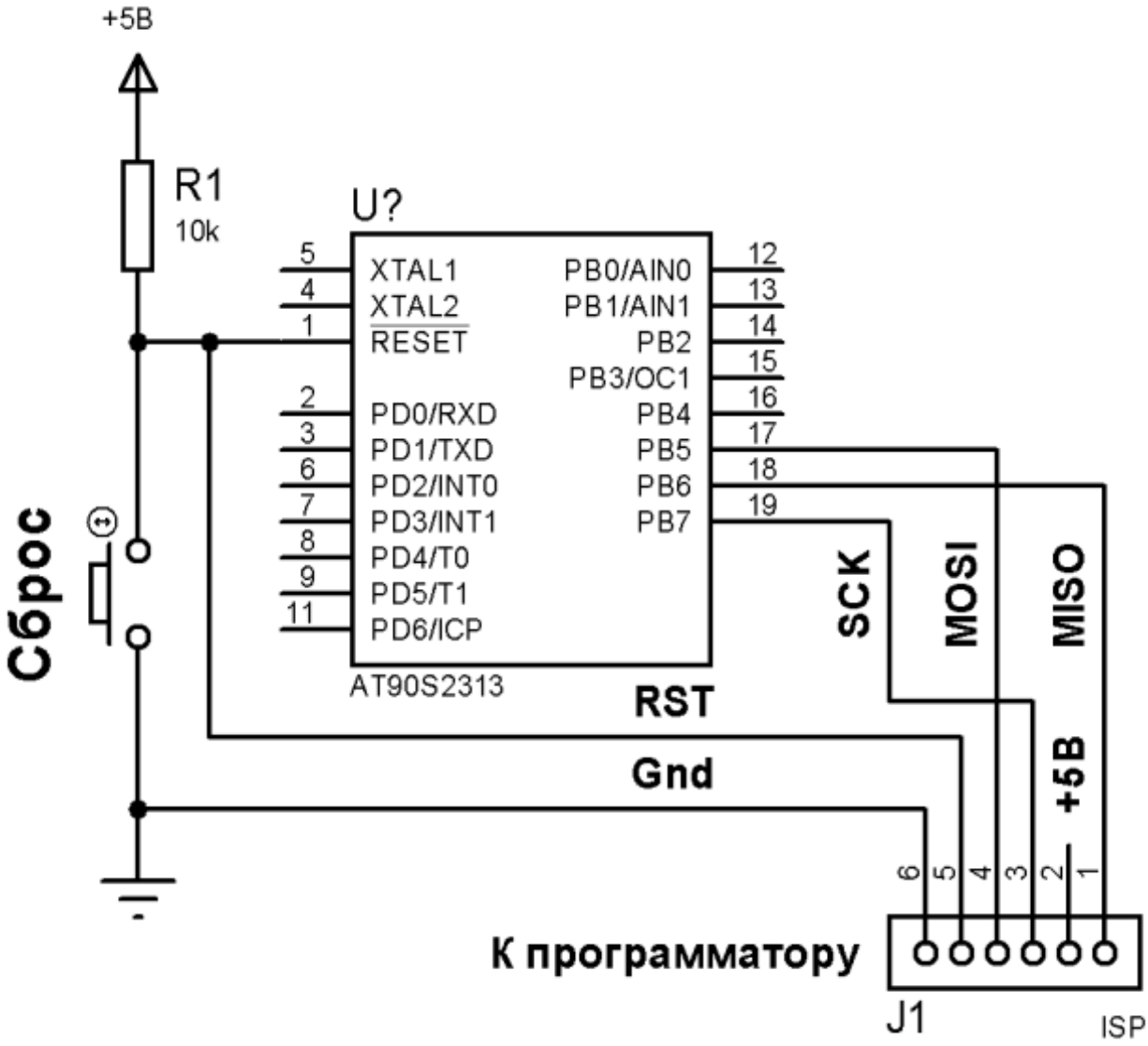


Схема подключения внутрисхемного программатора

Программа работы

1. Ознакомиться с примерами программ управления шаговым двигателем
2. Построить и отладить программу 1 (рис. 4)
3. Проверить работу программы 1 на модели в Proteus
4. Записать программу в память микроконтроллера и проверить ее работу
5. Построить и отладить программу 4
6. Проверить работу программы 4 на модели в Proteus
7. Записать программу в память микроконтроллера и проверить ее работу
8. Разработать, отладить и проверить работу программ для задач 1 и 2
9. Оформление отчета

Задача 1. Регулирование скорости

1. Задается начальное значение TN (временной промежуток между сменами фаз)
2. При пуске начальная скорость определяется TN
3. При поступлении «0» на PIND.2 уменьшается TN (при TN >0, TN--)
4. При поступлении «0» на PIND.3 увеличивается TN (при TN <255, TN++)
5. При сбросе ШД останавливается
6. ШД может останавливаться при PIND.2 = PIND.3 = 0, сохраняя установленную скорость при повторном пуске

Задача 2. Задание числа шагов поворота

1. Задается начальное значение TN (временной промежуток между сменами фаз) и число шагов Ns
2. При пуске скорость определяется TN
3. При поступлении «0» на PIND.2 выполняется Ns шагов влево
4. При поступлении «0» на PIND.3 выполняется Ns шагов вправо
5. При сбросе ШД останавливается
6. Для управления числом шагов необходимы дополнительные сигналы

Содержание отчета

1. Схема управления ШД (рис. 6) и подключения программатора
2. Программы 1 и 4
3. Программа для задачи 1
4. Программа для задачи 2
5. Выводы по работе

ЛАБОРАТОРНАЯ РАБОТА №4
ЖКИ МОДУЛИ СО ВСТРОЕННЫМ КОНТРОЛЛЕРОМ HT1611/3

Практически любое микроконтроллерное устройство имеет те или иные устройства индикации. В простейшем случае это всего несколько светодиодов, а порой это цветной графический дисплей. Появление модулей ЖКИ со встроенными контроллерами значительно упростило схемы сопряжения. Наиболее универсальными из таких модулей являются матричные алфавитно-цифровые, которые позволяют отображать цифры, буквы латинского и русского алфавита и даже псевдографику, используя возможности загружаемых символов. Однако такие ЖКИ - модули довольно дороги, они не отличаются малым энергопотреблением, да и в ряде устройств просто избыточны. Там, где не требуются широкие возможности устройств индикации, более подходящими могут оказаться 7-сегментные ЖКИ - модули.

Среди 7-сегментных ЖКИ - модулей наибольшее распространение получили модули на основе контроллера HT1611 (или HT1613). Они имеют 10 знакомест и управляются по последовательной шине.

Общие характеристики

Индикатор на жидких кристаллах (LCD).
Количество символов – 10
Размер модуля - 110x46x10 мм
Размер видимого поля - 72x27 мм
Высота символа (двух видов) - 10мм("маленький")-15мм("большой").
Напряжение питания - 1.2-1.7V
Потребляемый ток - 10mkA
Кроме функции контроллера в модуль включена поддержка часов реального времени и таймера с выводом данных на индикатор.

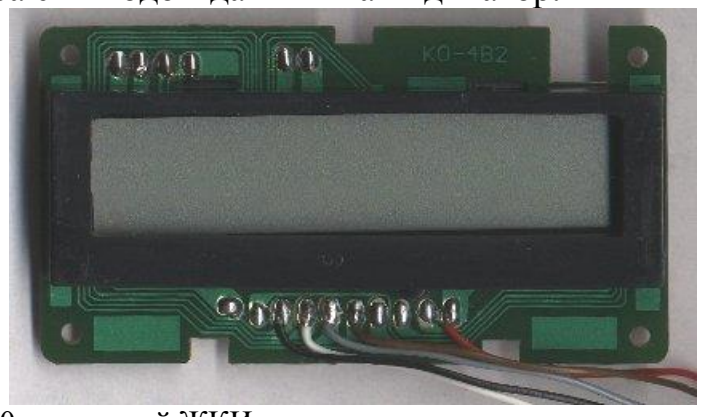


Рис. 1. 10-разрядный ЖКИ - модуль на основе контроллера HT1611.

Кроме отображения информации, передаваемой в модуль по шине, он может работать автономно в режиме часов реального времени. Для этого модуль имеет кварцевый резонатор и выводы для подключения кнопок установки времени.

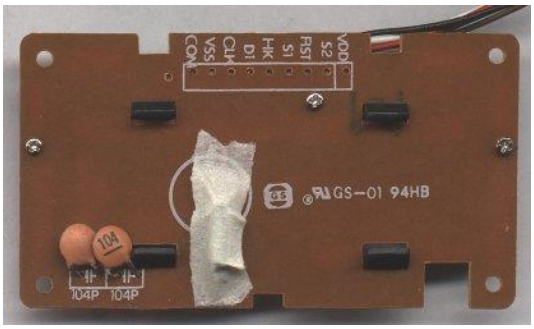


Рис. 2. Под липкой бумажной лентой расположен кварцевый резонатор.

Кроме того, возможна работа модуля в режиме таймера. Низкое напряжение питания (1.5 В) и малый ток потребления (не более 10 мка) легко позволяют организовать резервное питание. Однако то, что встроенные часы никак не связаны с последовательным интерфейсом, делает их очень ограниченными в использовании. Что поделаешь, индикатор в основном предназначен для телефонных аппаратов и оптимизирован именно для такого применения.

Назначение выводов модуля показано в таблице:

Номер вывода	Название вывода	Функция
1	12/24	переключение формата времени
2	Vss	общий
3	SK	тактовая линия шины
4	DI	линия данных шины
5	HK	переключение часы/индикатор
6	S1	установка времени
7	S2	выбор режима установки времени
8	TMR	сброс таймера
9	Vdd	напряжение питания

Примечание: выводы 1, 6, 7, 8 имеют внутреннее подключение к общему проводу через резисторы примерно 5 М. Выводы 3, 4, 5 имеют внутреннее подключение к выводу питания через резисторы примерно 1 М.

Режимом работы модуля управляет сигнал HK, который в телефонном аппарате соответствует сигналу поднятия трубки. Когда HK=1, модуль находится в режиме отображения реального времени. Если установить HK=0, то модуль переходит в режим таймера, начиная отсчет с нуля. Максимальный интервал, отсчитываемый таймером - 59 мин. 59 сек. Если в режиме таймера по шине приходят символы, индикатор очищается, и они начинают отображаться с крайней правой позиции. Новый поступивший символ вызывает сдвиг имеющихся символов влево. Если более 10 сек новые символы не поступают, модуль снова переходит в режим таймера, начиная отсчет с нуля. Если во время отображения поступивших по шине символов установить HK=1, то модуль вернется в режим отображения реального времени. Если же установить HK=1 во время индикации значения таймера, то возврат в режим отображения реального времени произойдет с задержкой 5 сек.

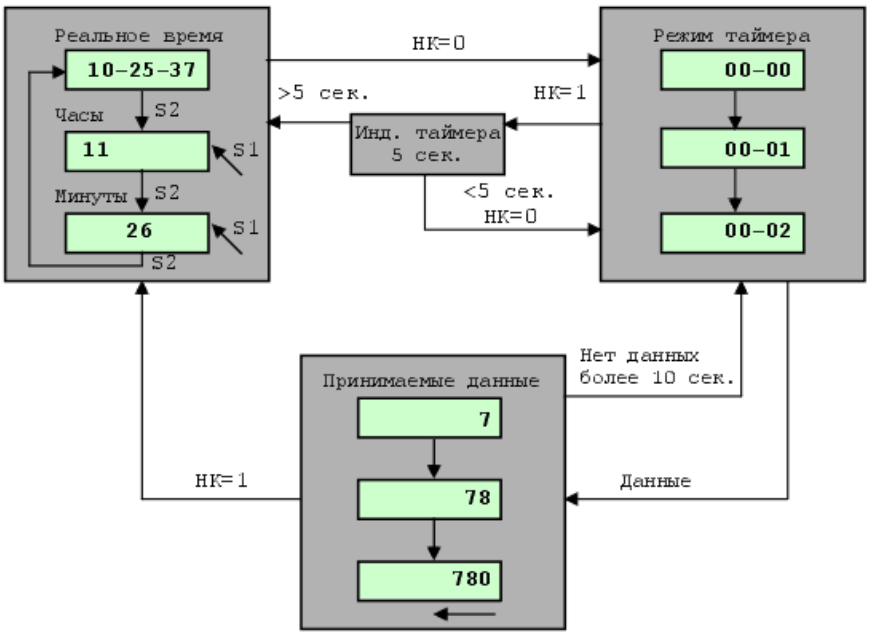


Рис. 3. Блок-схема алгоритма работы модуля.

К выводам модуля S1 и S2 могут быть подключены кнопки для установки текущего времени. Когда модуль находится в режиме отображения реального времени, нажатие кнопки S2 позволяет перейти в режим установки часов или минут (остальные цифры гаснут). При этом кнопкой S1 можно набрать требуемое значение. Затем с помощью кнопки S2 можно снова вернуться в режим отображения реального времени.

К выводу TMR может быть подключена кнопка сброса таймера. Когда модуль находится в режиме индикации значения таймера, первое нажатие этой кнопки сбрасывает таймер, и отсчет начинается с нуля. Повторное нажатие останавливает таймер (режим удержания показаний). Следующее нажатие снова запустит таймер с нулевого значения.

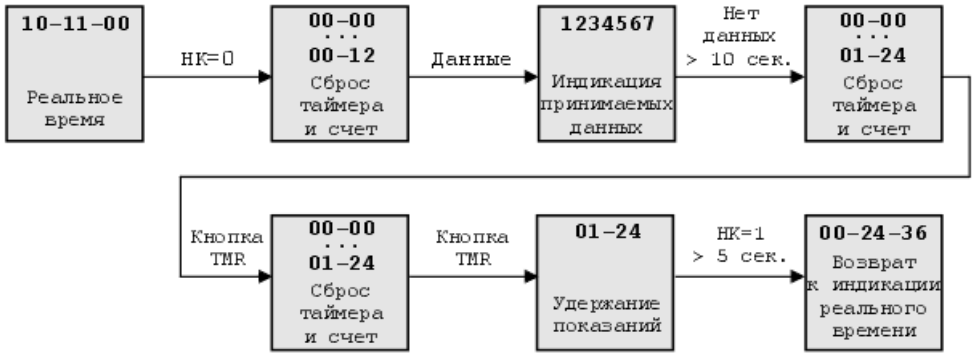


Рис. 4. Действие кнопки TMR.

Когда модуль находится в режиме отображения реального времени, нажатие кнопки TMR переводит его в режим секундомера. Как и в режиме таймера, кнопкой TMR можно сбросить его значение, запустив счет с нуля, или включить режим удержания показаний. Если в режиме удержания показаний модуль находится 5 сек, то происходит автоматический переход в режим отображения реального времени. Все кнопки подключаются между соответствующими выводами модуля и выводом питания.

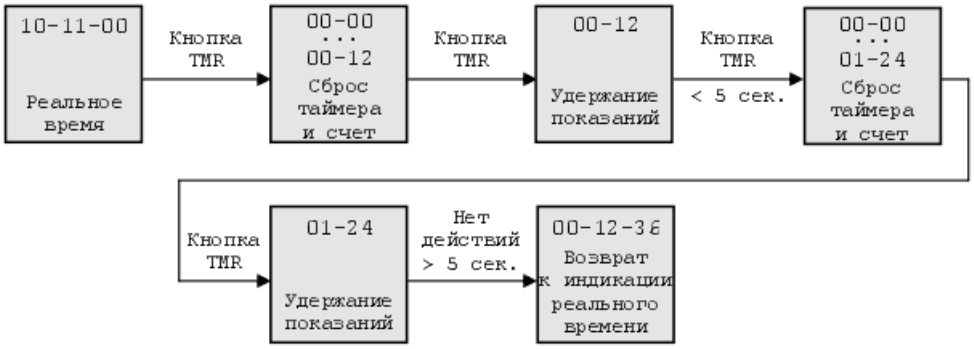


Рис. 5. Действие кнопки TMR в режиме индикации реального времени.

При использовании модуля в микроконтроллерной системе только для отображения загружаемых по последовательной шине символов, требуется соединить вывод HK с общим проводом, а выводы 12/24, S1, S2 и TMR оставить свободными. Временная диаграмма передачи данных по последовательной шине приведена на рисунке, где ta - время установки данных (>1 мкс), tb - время удержания данных (>2 мкс), tc - интервал между символами (>5 мкс).

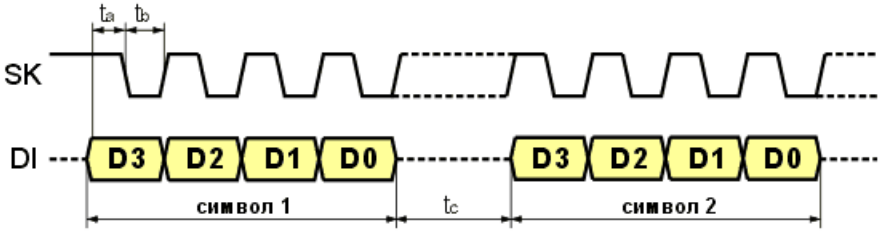


Рис. 6. Временная диаграмма передачи данных.

Данные подаются на линию DI и защелкиваются по спаду тактовых импульсов на линии SK. Символы отображаются в крайней правой позиции, уже имеющиеся на индикаторе символы сдвигаются влево. После того, как все необходимые данные переданы, линию SK следует оставить в состоянии низкого логического уровня, чтобы предотвратить автоматический переход модуля в режим отображения значения таймера.

Каждый символ кодируется 4-мя битами, поэтому всего имеется 16 символов.

D3	D2	D1	D0	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7

D3	D2	D1	D0	
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	b
1	1	0	0	C
1	1	0	1	d
1	1	1	0	E
1	1	1	1	F

Рис. 7. Таблица знакогенератора.

Нужно отметить, что напряжение питания индикатора сильно влияет на контрастность. При низком напряжении контрастность недостаточна, а при большом засвечиваются погашенные сегменты. Оптимум находится в промежутке 1.50 ... 1.65 В. Распространенная схема питания, где в качестве источника образцового напряжения используются диоды в прямом включении (рисунок a), не позволяет получить оптимальную контрастность, так как двух диодов оказывается мало, а трех - много. Тем более, желательно иметь возможность регулировки этого напряжения. Простая схема на одном транзисторе позволяет получить нужное напряжение питания и регулировать его (рисунок b).

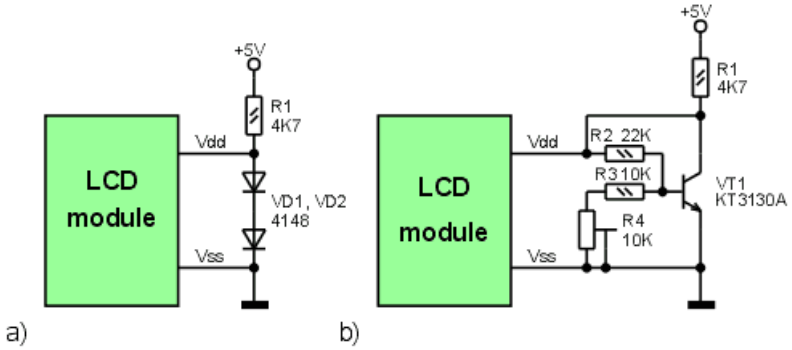


Рис. 8. Различные схемы питания индикатора.

Учитывая очень низкий ток потребления индикатора можно обойтись и простым резисторным делителем, если входное напряжение питания постоянно. Описанные схемы питания не являются экономичными и подходят, например, для устройств с сетевым питанием. Система питания автономного устройства может быть очень сложной, и конкретные решения зависят от специфики задачи. Одним из вариантов может быть питание устройства от элемента напряжением 1.5 В, от которого индикатор питается непосредственно. Микроконтроллерная часть устройства питается от того же элемента через повышающий DC-DC преобразователь.

Для согласования логических уровней можно применить разные схемы. Учитывая тот факт, что входы DI и SK имеют внутренние подтягивающие резисторы, можно обойтись просто диодами (рисунок a). Преимущество такого способа заключается в том, что согласование не будет зависеть от напряжения питания микроконтроллера. Однако такой способ имеет и недостаток. Ввиду больших номиналов подтягивающих резисторов уровни на входах будут довольно медленно достигать состояния логической единицы, что потребует значительного снижения скорости обмена. Поэтому предпочтительнее для согласования использовать резисторные делители (рисунок b).

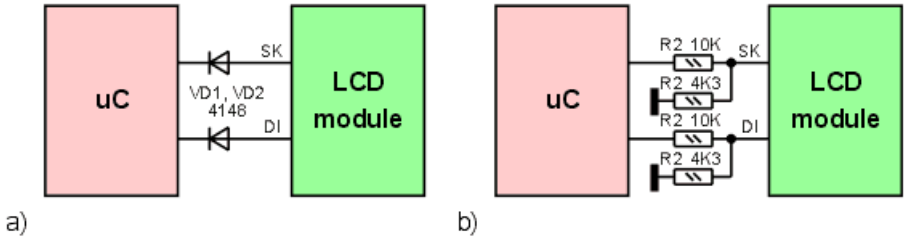


Рис. 9. Согласование логических уровней.

Необходимо отметить, что в течение примерно 2 сек после включения питания модуль не воспринимает данные, передаваемые ему по последовательной шине. Поэтому всегда должна быть задержка между включением питания и началом обмена.

Описанный модуль, тем не менее, обладает рядом существенных недостатков:

- отсутствие десятичных точек; это затрудняет отображение произвольных цифровых величин
- невозможность управления отдельными сегментами; контроллер имеет фиксированный знакогенератор, содержащий всего 16 символов, в то же время возможности 7-сегментного индикатора значительно шире, он может отображать многие буквы латинского алфавита и специальные символы
- заметные мерцания изображения при загрузке, вызванные отсутствием буферизации выводимой информации; при необходимости обновления изображения с большой частотой создается весьма неприятный зрительный эффект
- высокая инерционность, что ограничивает применение таких эффектов, как, например, мигающие символы
- плохой угол обзора; индикатор оптимизирован для применения в телефонных аппаратах, где он устанавливается на слегка наклонной панели; при установке индикатора, например, на вертикальную переднюю панель прибора контрастность изображения резко падает
- отсутствие возможности регулировки контрастности; такую регулировку можно осуществить только изменением напряжения питания, что не всегда удобно
- отсутствие встроенной подсветки; место для ее установки не предусмотрено
- требуются дополнительные элементы для получения напряжение питания 1.5 В и для согласования логических уровней, так как напряжение питания микропроцессорной системы обычно выше.

Программирование вывода символа на индикатор HT161x

В соответствии с описание HT161x построим временную диаграмму вывода символа «5», которому соответствует код 0101 (рисунок 10). Метки 0,1,2, … ,11,12 на оси времени диаграммы соответствуют моментам изменения комбинации сигналов DI, SK управления индикатором.

Таким образом, для вывода символа «5» необходимо последовательно вывести на линии порта МК, связанные с DI, SK комбинации : 00,01,00,10,11,10,00,01,00,10,11,10,00. Эти состояния выходов должны удерживаться в течение временных интервалов, соответствующих временным параметрам интерфейса индикатора.

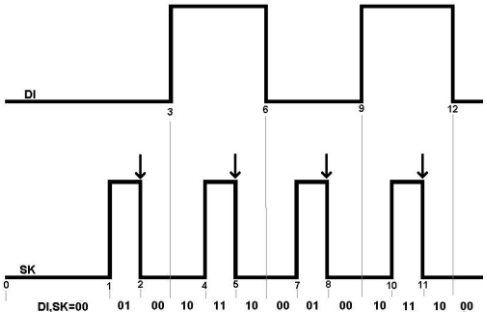


Рисунок 10. Временные диаграммы вывода символа «5»

Стрелками на временной диаграмме выделены моменты времени, когда контроллер индикатора осуществляет ввод битов кода символа (биты передаются, начиная со старшего бита –D3). В эти моменты сигнал на линии данных (DI) не должен изменяться. С учетом этого, состояния, соответствующие интервалам 3-4, 6-7, 9-10 можно исключить для сокращения числа операций вывода в порт до: 00,01,00,11,10,01,00,11,10,00 (рис. 11).

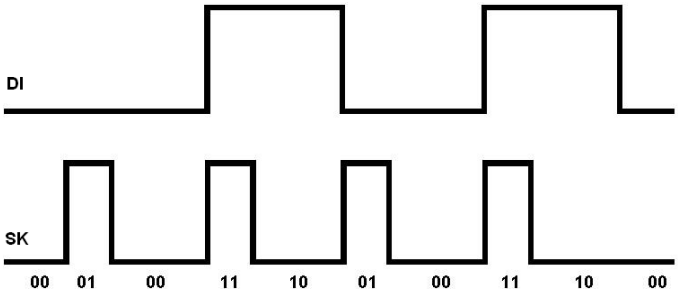


Рисунок 11. Сокращенная последовательность операций вывода

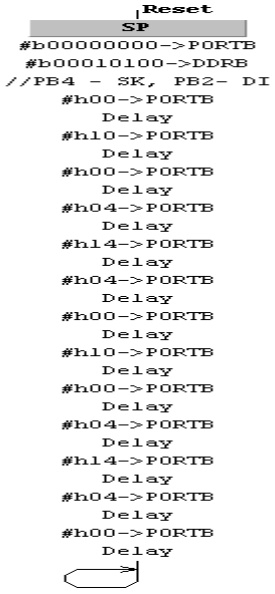
С сигналами DI, SK могут быть связаны произвольные линии портов и даже разных портов. Пусть DI связан с PORTB.2, а SK с PORTB.4. Если другие линии не используются для вывода, то приняв для них значение 0, получим последовательность кодовых комбинаций (таблица 1) для байтового вывода.

Таблица 1. Кодовые комбинации для вывода в порт

№	DI	SK	PORTB								BIN	HEX
			7	6	5	**SK**	3	**DI**	1	0	код	код
1	0	0	0	0	0	**0**	0	**0**	0	0	#b00000000	#h00
2	0	1	0	0	0	**1**	0	**0**	0	0	#b00010000	#h10
3	0	0	0	0	0	**0**	0	**0**	0	0	#b00000000	#h00
4	1	0	0	0	0	**0**	0	**1**	0	0	#b00010000	#h04
5	1	1	0	0	0	**1**	0	**1**	0	0	#b00010100	#h14
6	1	0	0	0	0	**0**	0	**1**	0	0	#b00010000	#h04
7	0	0	0	0	0	**0**	0	**0**	0	0	#b00000000	#h00
8	0	1	0	0	0	**1**	0	**0**	0	0	#b00000100	#h10
9	0	0	0	0	0	**0**	0	**0**	0	0	#b00000000	#h00
10	1	0	0	0	0	**0**	0	**1**	0	0	#b00010000	#h04
11	1	1	0	0	0	**1**	0	**1**	0	0	#b00010100	#h14
12	1	0	0	0	0	**0**	0	**1**	0	0	#b00010000	#h04
13	0	0	0	0	0	**0**	0	**0**	0	0	#b00000000	#h00

Последовательность вывода кодов и программа выглядят следующим образом:

#b00000000->PORTB или #h00->PORTB
#b00010000->PORTB или #h10->PORTB
#b00000000->PORTB или #h00->PORTB
#b00000100->PORTB или #h04->PORTB
#b00010100->PORTB или #h14->PORTB
#b00000100->PORTB или #h04->PORTB
#b00000000->PORTB или #h00->PORTB
#b00010000->PORTB или #h10->PORTB
#b00000000->PORTB или #h00->PORTB
#b00000100->PORTB или #h04->PORTB
#b00010100->PORTB или #h14->PORTB
#b00000100->PORTB или #h04->PORTB
#b00000000->PORTB или #h00->PORTB



Подпрограмма задержки Delay выполняется в течение 61 такта синхронизации (61 мкс при Fosc =1 Мгц), что достаточно для получения максимальной для HT161x задержки 5 мкс в диапазоне Fosc= 1 .. 12 Мгц.

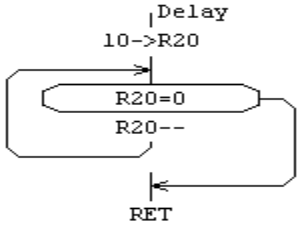


Рисунок 12. Программа вывода символа «5»

Размер программы – 63 слова, что составляет менее 7% объема Flash-памяти микроконтроллера AT90S2313 (ATtiny2313). Моделирование процесса вывода на индикатор посредством приведенной выше программы дает результат, приведенный на рис. 13. Развертка на виртуальном осциллографе - 200 мкс/деление.

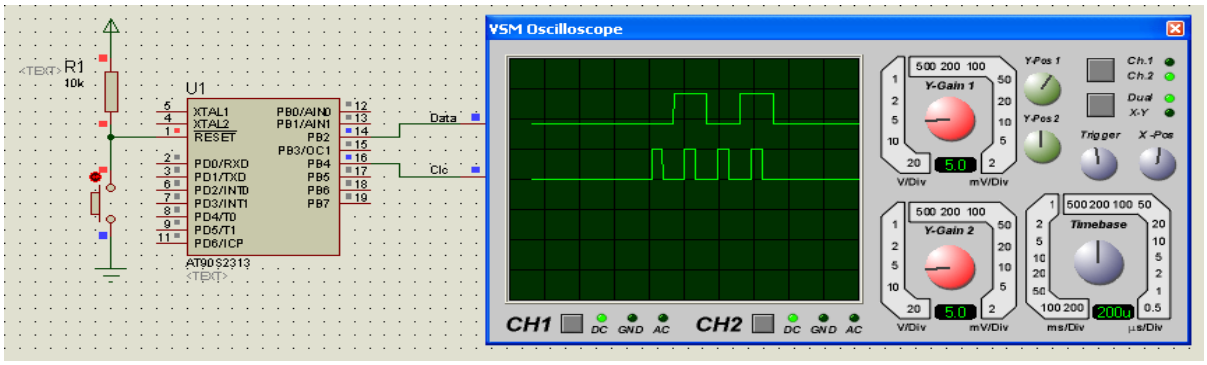


Рисунок 13. Моделирование вывода символа «5»

Если использовать битовые операции вывода и подпрограмму формирования импульсов синхронизации Pulse с необходимыми задержками, то получим программу (рис. 5), которая занимает 38 слов.

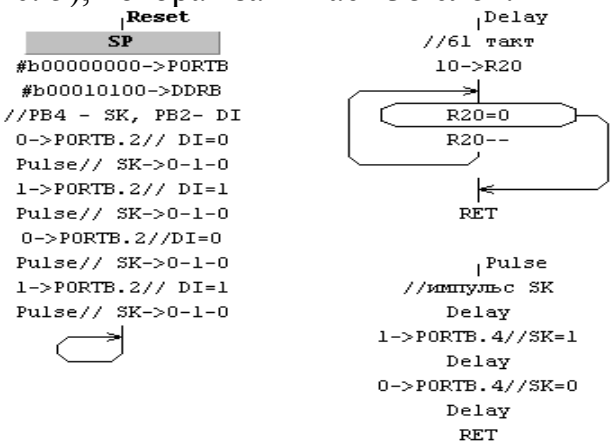


Рисунок 14. Программа с использованием битовых операций вывода

Программа вывода строки «0123456789», благодаря использованию подпрограммы вывода символа с передачей кода символа через R24, занимает 47 слов.

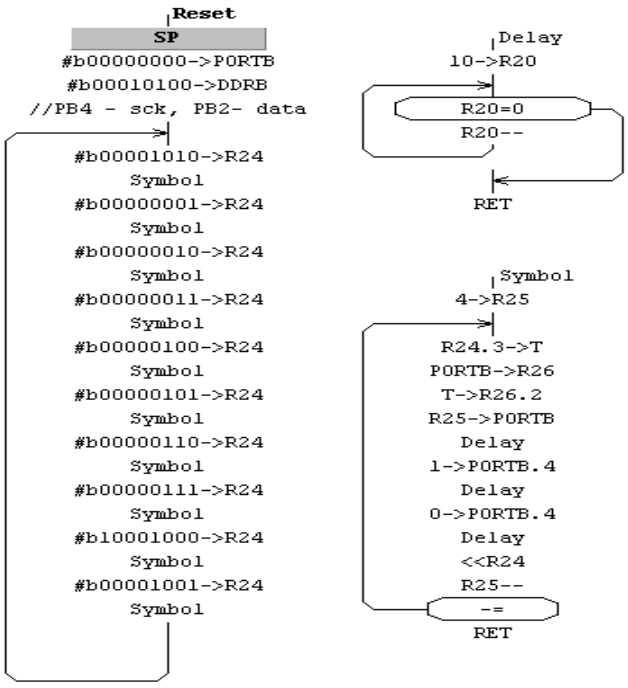


Рисунок 15. Программа вывода строки символов

ЛАБОРАТОРНАЯ РАБОТА №5
ИССЛЕДОВАНИЕ ТАЙМЕРОВ-СЧЕТЧИКОВ AVR МК

Программа работы:

1. Ознакомиться с элементами программной модели таймеров-счетчиков TC0, TC1 [1] микроконтроллера AT902313 (рис. 1)

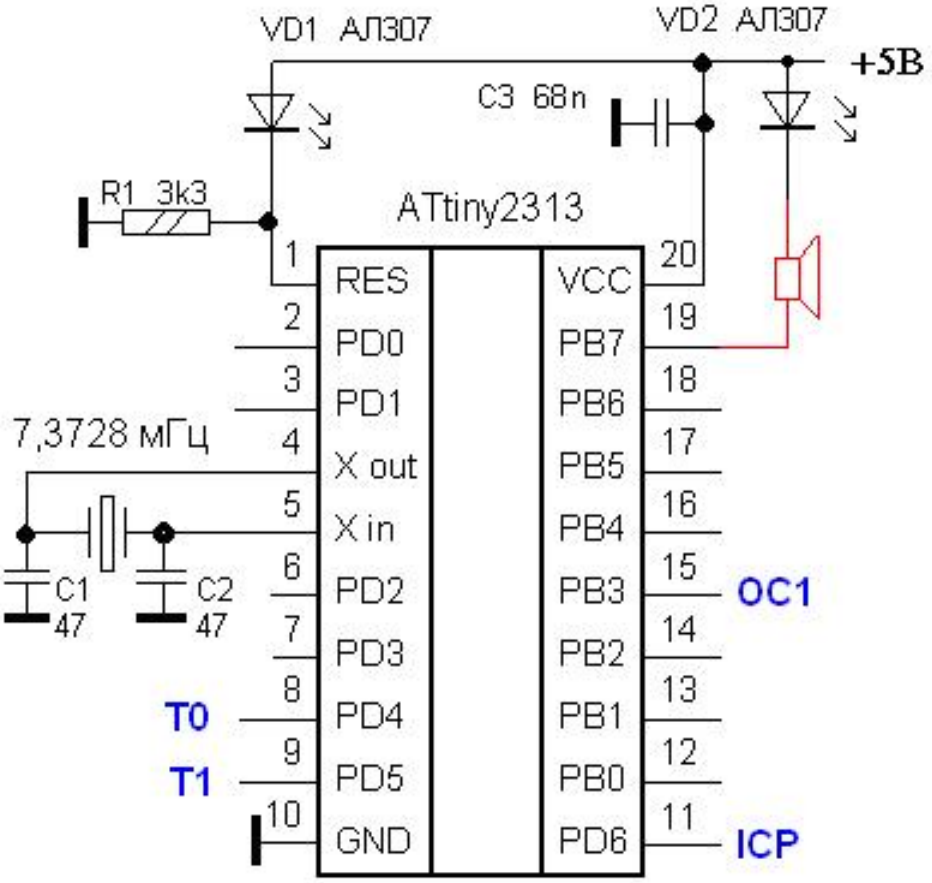


Рисунок 1. Линии микроконтроллера, связанные с таймерами–счетчиками TC0 (8 разрядов) и TC1 (16 разрядов).

2. Исследование работы TC0 в режимах счета перепадов счетных импульсов

Упрощенная схема устройства на основе микроконтроллера AT90S2313 (рис.2) содержит источник счетных импульсов – кнопку «Счет» и светодиодный «столбиковый» индикатор для отображении состояния линий порта PB7..0. Светодиоды светятся при подаче на них «1» (высокого уровня напряжения). При нажатии и отпускании кнопки формируется импульс на входе T0 и на выходе МК код должен увеличиться ($19=00010011_2$ -> $00010100_2=20$). После кода $11111111_2=255$ следует код 00000000_2 и цикл счета повторяется.

При нажатии «Сброс» устанавливается состояние выхода 00000000_2.

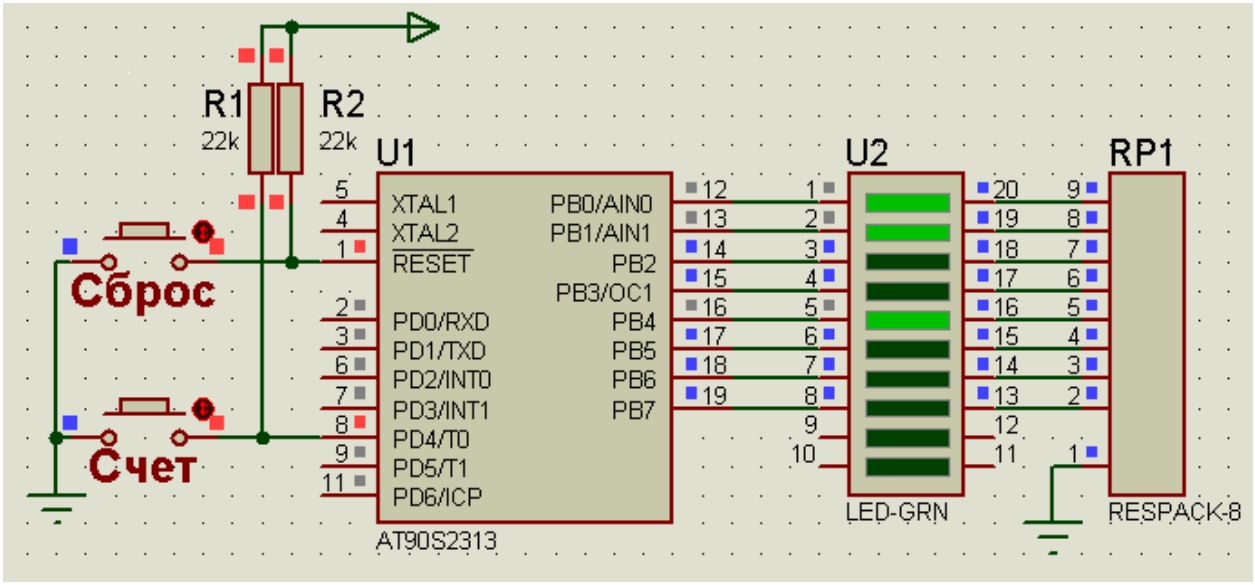


Рисунок 2. Схема счетчика внешних импульсов T0
с индикацией в двоичном коде (текущий код 00010011_2 =19)

При построении программы необходимо установить состояние регистра TCCR0 для режима счета записью соответствующей константы, либо посредством элементов “Setter”.

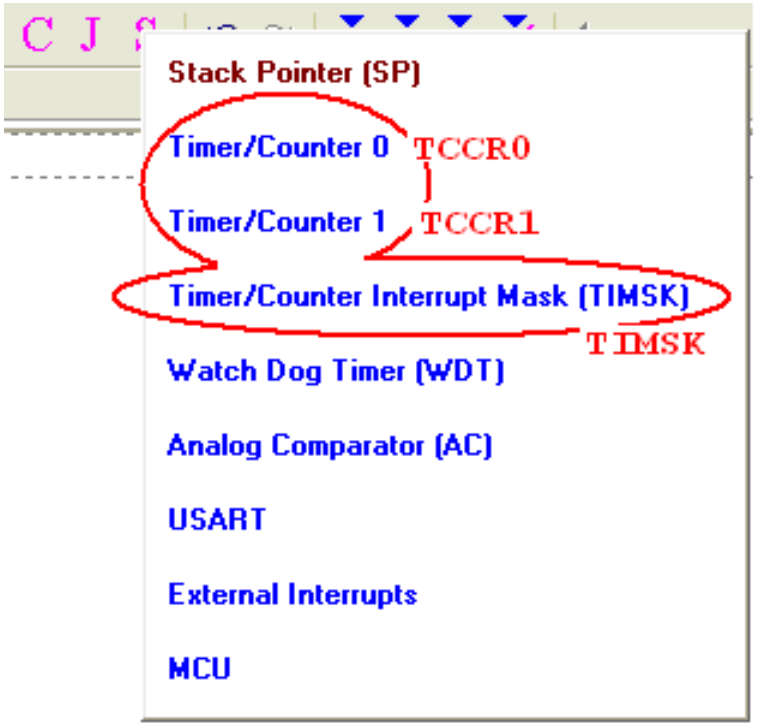


Рисунок 3. Меню выбора элементов “Setter” для регистров таймеров

В диалоговом окне (рис. 4) выбирается режим счета перепадов сигнала T0. При установке режима “failing edge” состояние счетчика увеличивается при нажатии кнопки (1->0), а в режиме “rising edge” состояние счетчика увеличивается при отпускании кнопки (0->1).

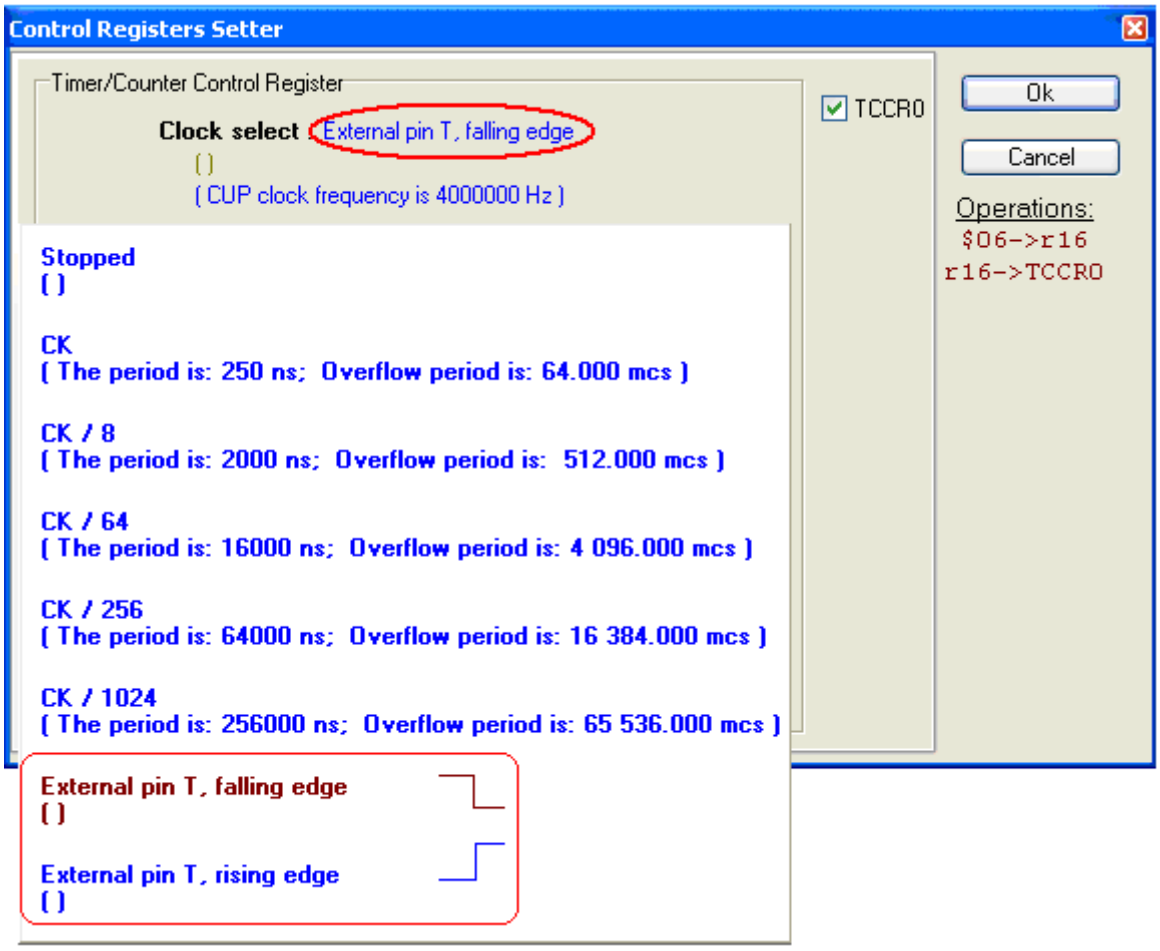


Рисунок 4. Настройка таймера- счетчика в режиме счета

Программа **TC0_count.alp** (рис. 5) содержит настройку TC0 и PORTB, а также бесконечный цикл пересылки содержимого счетчика TCNT0 в порт PORTB. Любое изменение состояния счетчика с минимальной задержкой появляется на линиях PB7..0 порта.

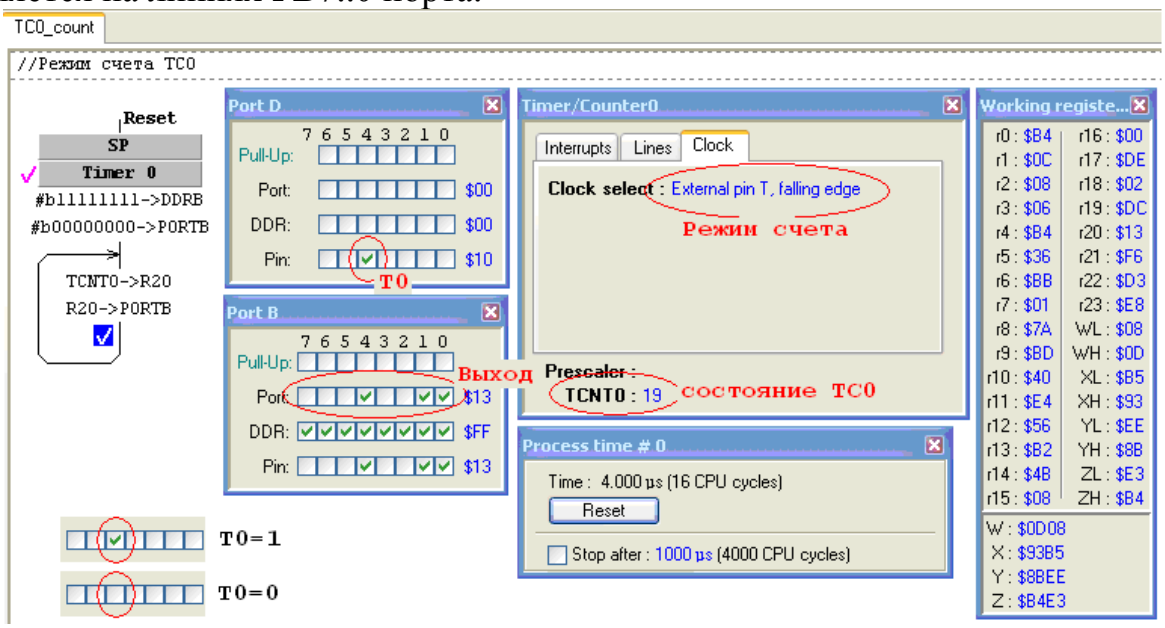


Рисунок 5. Алгоритм **TC0_count.alp** с использованием счетчика TC0

3. Исследование алгоритма обработки прерываний таймера TC0
При каждом прерывании должно изменяться состояние PB7.

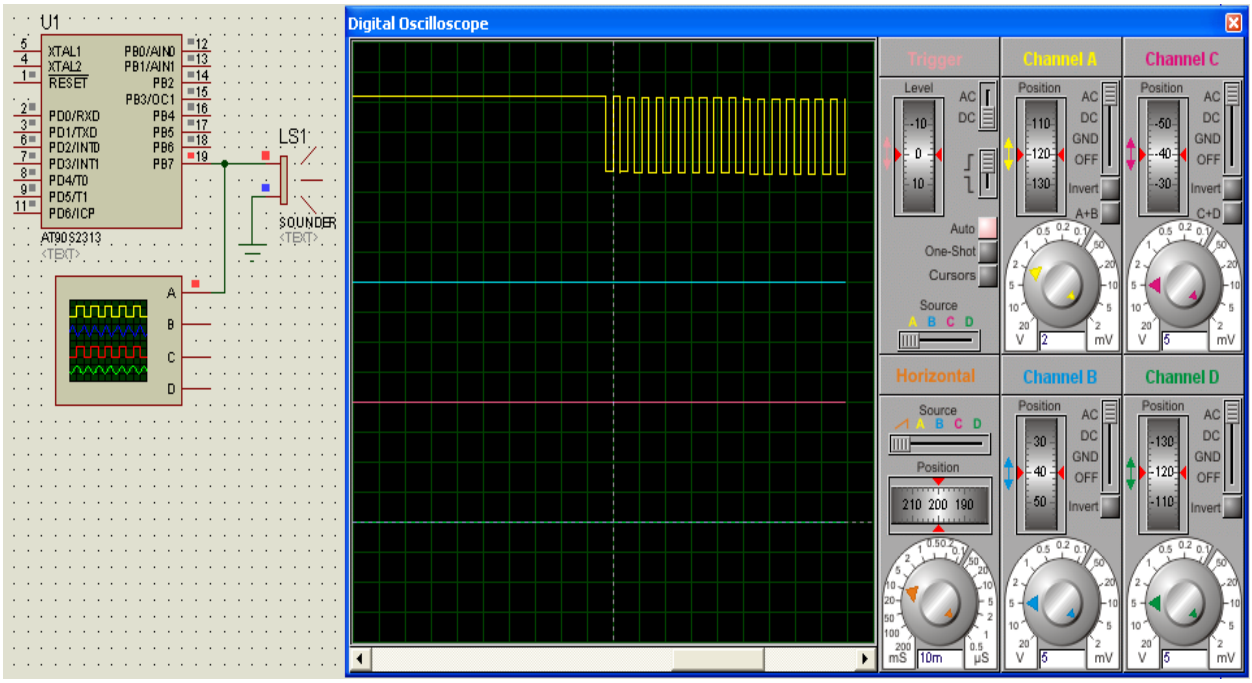


Рисунок 6. Схема эксперимента

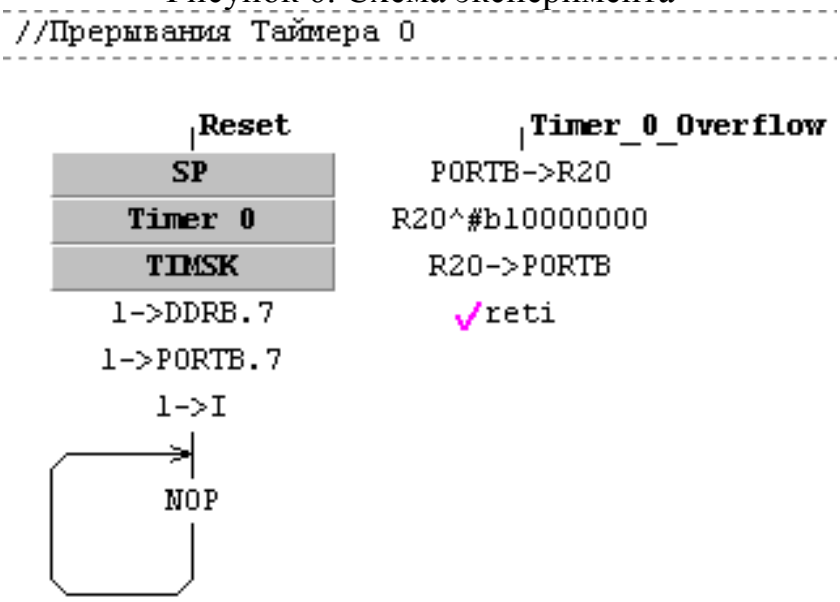


Рисунок 7. Алгоритм обработки прерываний TC0

Регистры TCO должны быть настроены следующим образом

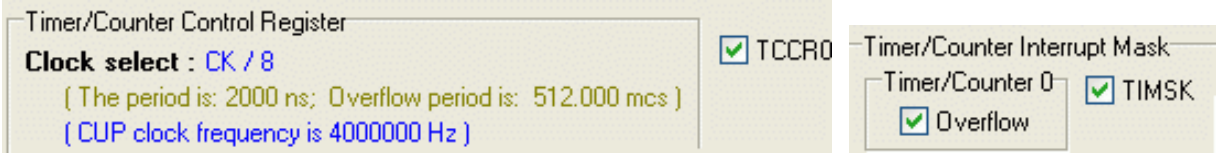
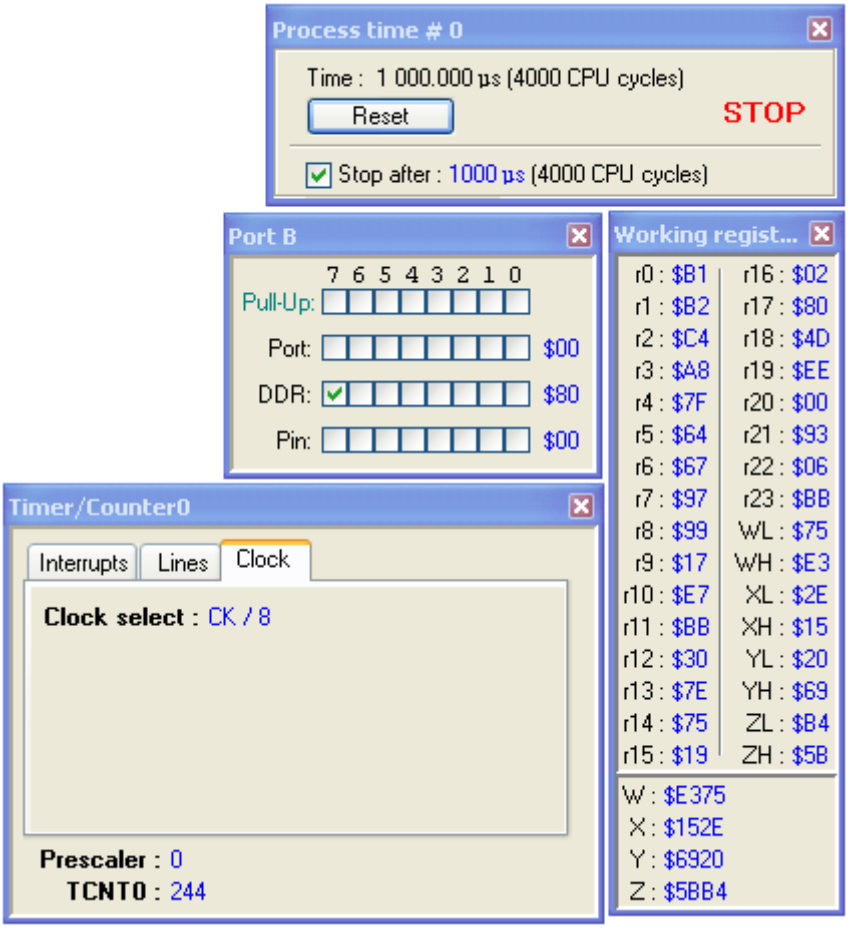


Рисунок 8. Настройка прерываний TC0



Программная модель в отладчике

4. Исследование алгоритма вызова прерываний с заданным периодом (рис.5). Период определяется кодом T0 предустановки TC0.

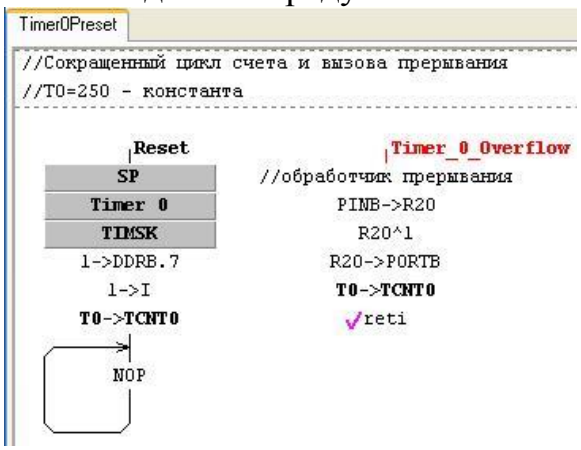


Рисунок 5. Алгоритм с предустановкой TC0

Определить период для T0=250. Для индивидуально заданного периода прерываний найти T0.

5. Совместное использование TC0, TC1 для воспроизведения последовательности звуков через PB7.

TC0 – отсчитывает полупериоды нот

TC1 – отсчитывает время звучания нот

Примеры проектирования МК-устройств и систем

Устройство формирования звуковых сигналов

При помощи МК можно достаточно просто воспроизводить (генерировать) различные звуковые сигналы, причем не только "бип-сигналы", но и разнообразные музыкальные фразы. Воспроизведение мелодии сводится к генерации последовательностей импульсных сигналов определенной звуковой частоты (нот) в течение определенных интервалов времени. Звуковой сигнал характеризуется двумя параметрами: частотой и длительностью. В табл. 8.1 приводятся значения частот для 12 нот, начиная с ноты ЛЯ первой октавы. Отношение соседних частот для равномерно темперированного музыкального строя выражается иррациональным числом $\sqrt[12]{2}$. От октавы к октаве частота звукового сигнала меняется ровно в 2 раза.

Таблица 8.1. **Значения частот основных нот**

Нота	Частота, Гц	Длительность полупериода, мкс	Код ноты
$ЛЯ_1$	440	1136	0
$ЛЯ_1$ #	466,2	1073	1
$СИ_1$	494	1012	2
$ДО_2$	523	956	3
$ДО_2$ #	554,36	902	4
$РЕ_2$	588	850	5
$РЕ_2$ #	622.25	804	6
$МИ_2$	660	758	7
$ФА_2$	698	716	8
$ФЛ_2$ #	739,99	677	9
$СОЛЬ_{\flat 2}$	784	638	A
$СОЛЬ_{\flat 2}$ #	830,65	602	B

Таким образом, генерация одной ноты в музыкальной фразе может быть осуществлена двумя рассмотренными ранее процедурами: выдачей импульсного сигнала и задержкой. Допустим, что в наборе воспроизводимых микроконтроллером музыкальных фраз длительность звучания ноты может принимать только значения 1, 1/2, 1/4 и 1/8 с.

Кодированная запись музыкальной фразы может быть, например, такой, как показано на рис. 8.1. Схема процедуры генерации музыкальной фразы (MUSIC) представлена на рис. 8.2. Вначале производится загрузка регистра-указателя (DPTR) начальным адресом закодированной нотной записи. Затем выполняется настройка обоих таймеров на 16-битный формат, режим 1. Далее производится обращение к нотной записи, которая располагается в ВПД (для нашего примера), за очередным кодом. После этого проверяется на нуль прочитанный код, т.е. анализируется, достигнут или нет конец записи. При достижении конца нотной записи осуществляется выход из процедуры. Если же код не равен нулю, то подготавливаются данные для процедуры генерации ноты (SOUND). Длительность интервала звучания ноты при этом удваивается, так как процедура SOUND выдерживает интервал, равный (1/16) D с. Далее включается в работу процедура генерации ноты, после чего осуществляется переход к следующему коду нотной записи.

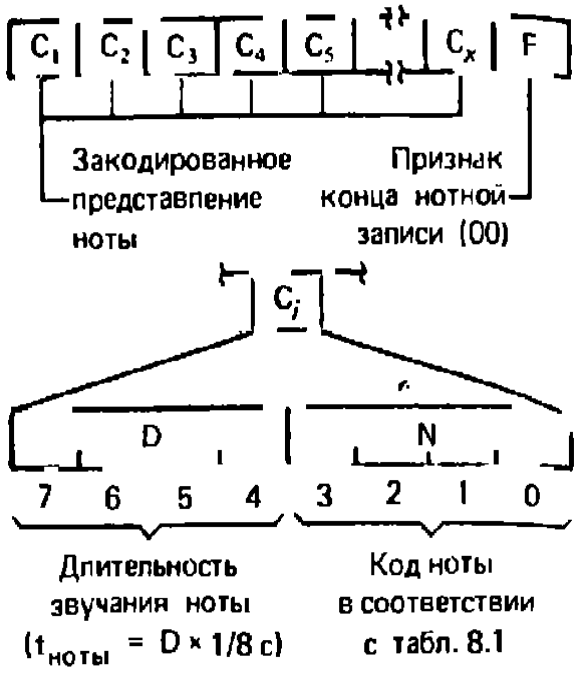


Рис. 8.1. Принятый вариант кодирования нотной записи

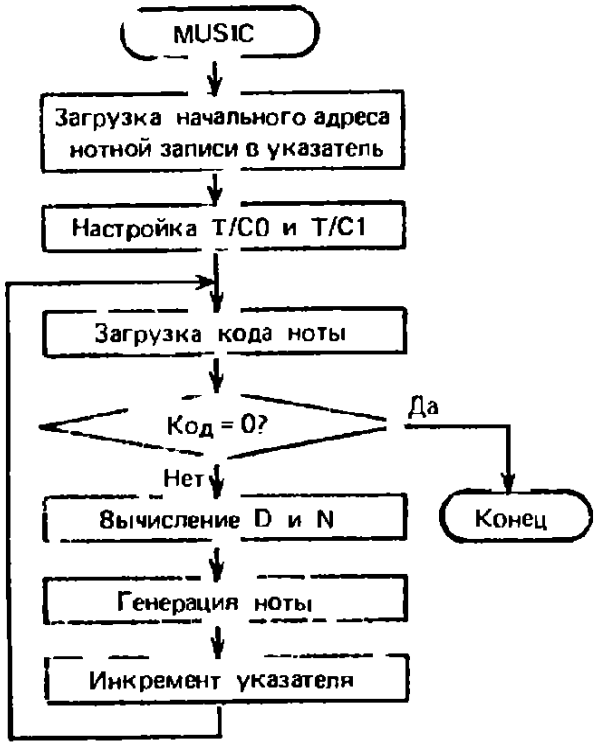


Рис. 8.2. Блок-схема алгоритма генерации музыкальной фразы

Схема процедуры генерации ноты приведена на рис. 8.3. Она ориентирована на МК51, так как использует два таймера: T/C1 – для формирования частоты сигнала и T/C0 – для формирования интервала звучания ноты. Звуковой сигнал формируется на выводе P1.0 путем его инвертирования каждые 0,5T, где T – период звучания ноты. Задержка длительностью 0,5T реализуется на T/C1 (к выходу P1.0 через усилитель подключен громкоговоритель).

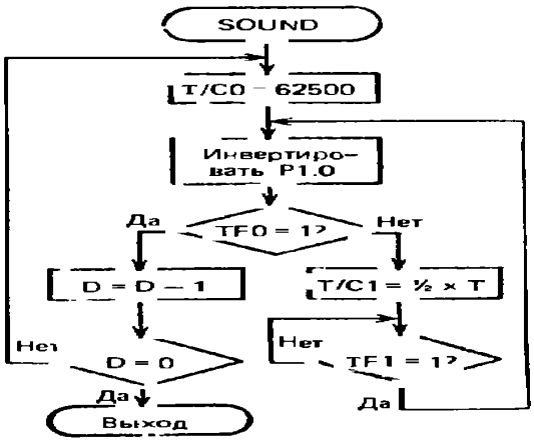


Рис. 8.3. Схема процедуры генерации ноты

Вначале осуществляется загрузка в T/C0 числа 62 500, что обеспечивает получение задержки в 1/16 с. При переполнении T/C0 производится уменьшение счетчика интервала D. При достижении условия D = 0 генерация прекращается.

Процедура SOUND имеет два входных параметра: длительность интервала генерации ноты D, измеряемая в единицах, равных 1/16 с (регистр R7), и удвоенный код ноты N (регистр R6).

Текст программы генерации музыкальной фразы, ориентированной на МК51, приводится ниже. Программа снабжена подробными комментариями. Для определения длительности полупериода сигнала соответствующей ноты используется метод табличного преобразования. В таблице с именем TAB хранятся значения полупериодов соответствующих нот в микросекундах, уменьшенные на 16. Такое уменьшение необходимо, так как с момента инвертирования сигнала на выходе P1.0 и до загрузки T/C1 проходит имено 16 мкс, необходимые для выполнения команд, расположенных между метками S1 и S0.

Если в конкретном применении МК закодированную нотную запись оказывается удобнее хранить в постоянной памяти программ, то программу придется немного модифицировать. Такую модификацию предлагается выполнить читателю самостоятельно.

Закодированная запись новогодней английской песенки, нотная запись которой представлена на рис. 8.4, занимает 50 байтов (49 нот и признак конца записи) и выглядит следующим образом:

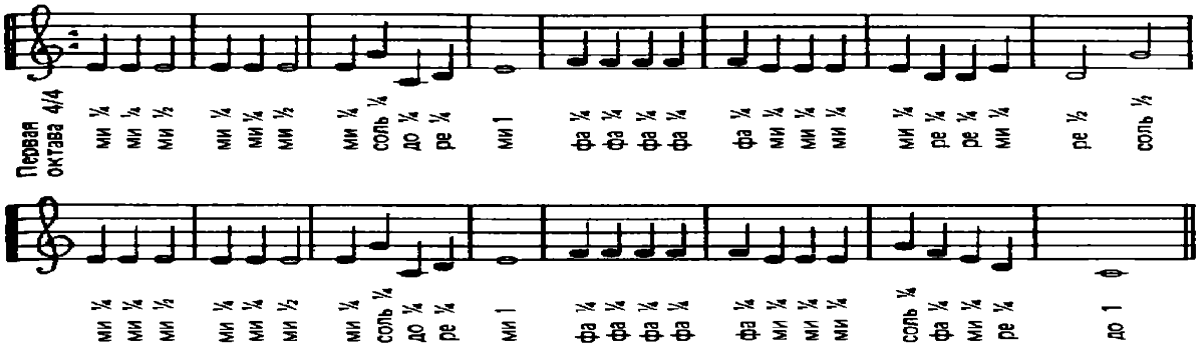
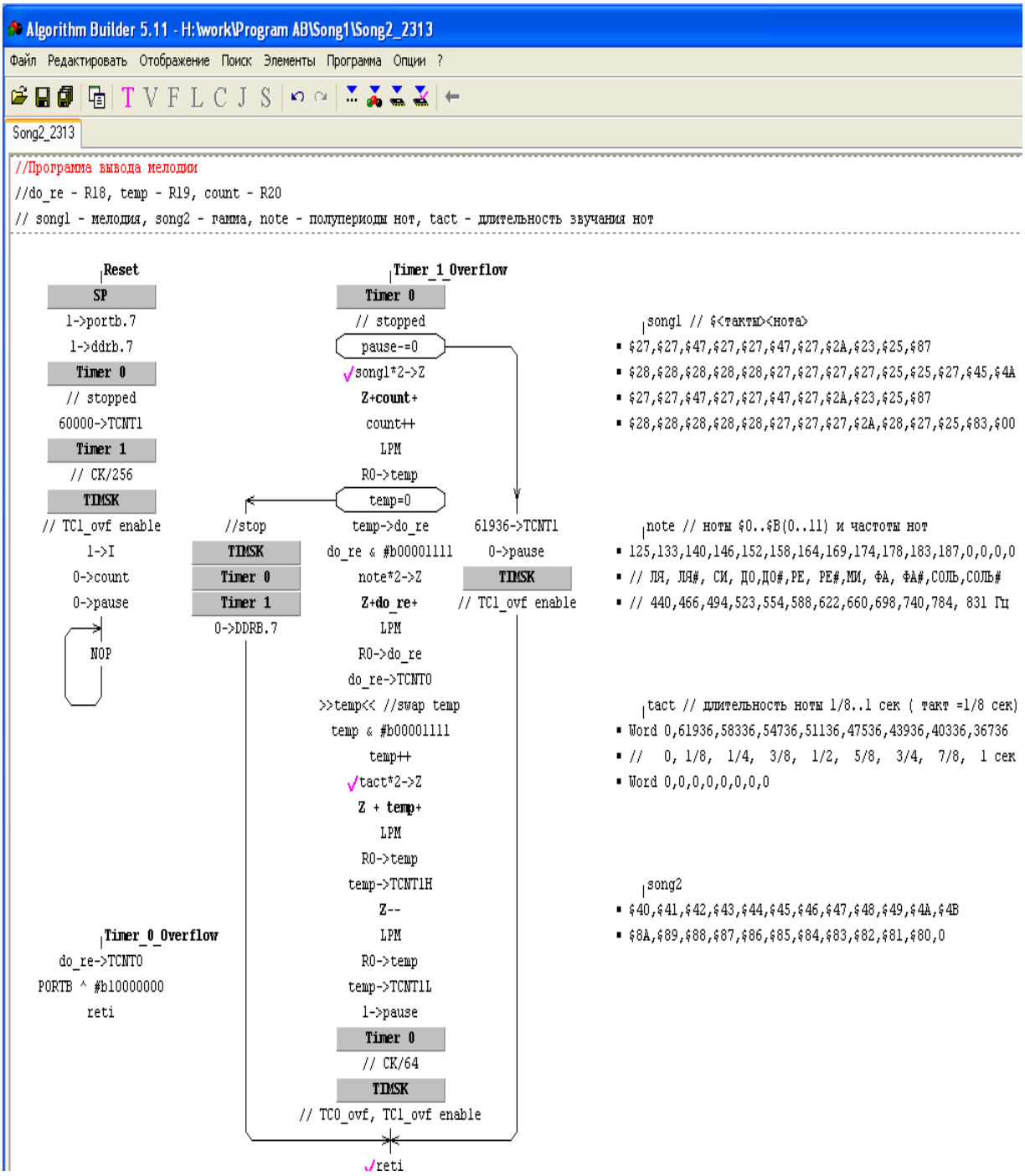


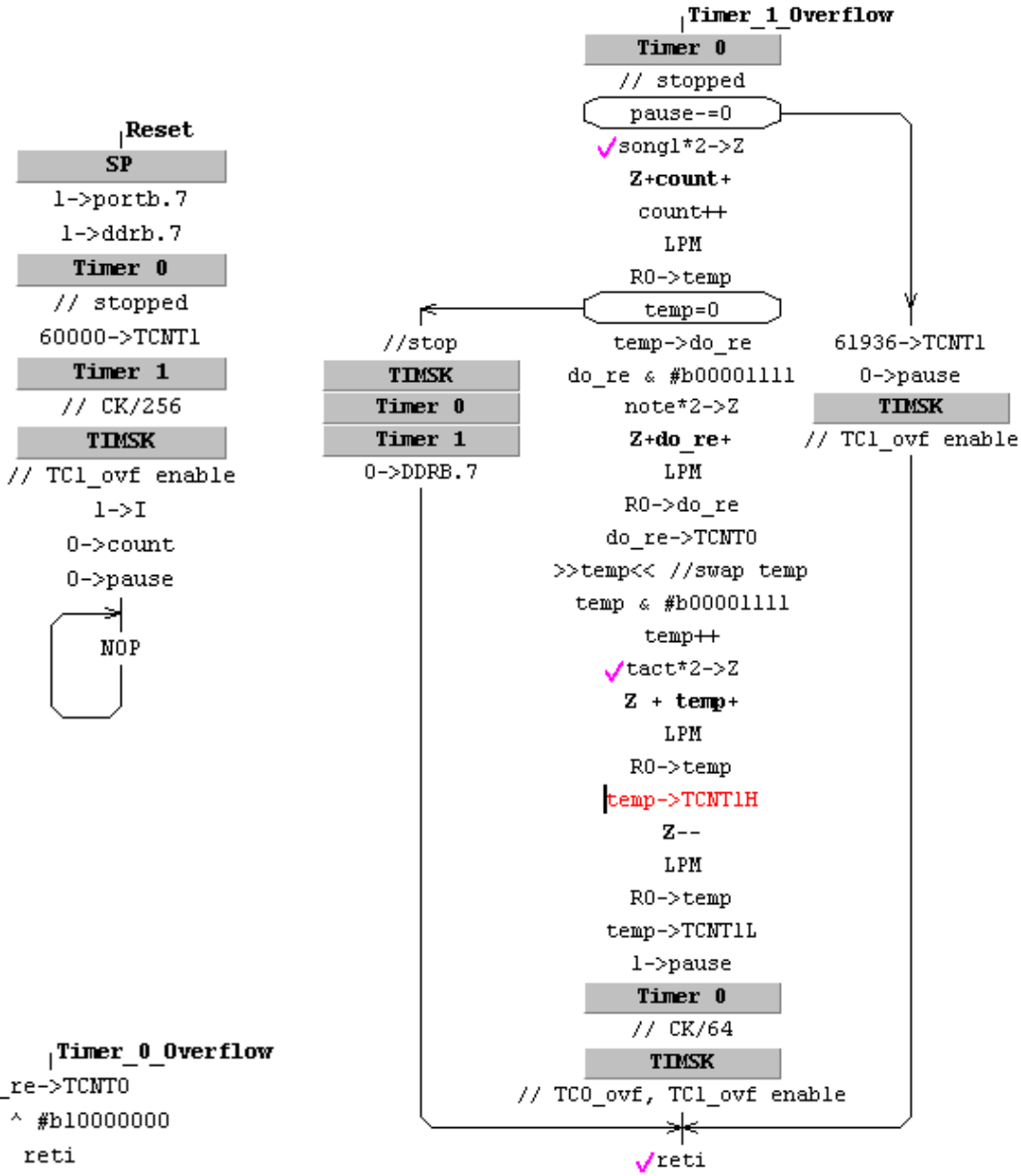
Рис. 8.4. Нотная запись песенки

```
songl // $<такты><нота>
$27,$27,$47,$27,$27,$47,$27,$2A,$23,$25,$87
$28,$28,$28,$28,$28,$27,$27,$27,$27,$25,$25,$27,$45,$4A
$27,$27,$47,$27,$27,$47,$27,$2A,$23,$25,$87
$28,$28,$28,$28,$28,$27,$27,$27,$2A,$28,$27,$25,$83,$00
```

Программа воспроизведения мелодии Song2_2313.alp




```
//Программа вывода мелодии
//do_re - R18, temp - R19, count - R20
// song1 - мелодия, song2 - гамма, note - полупериоды нот, tact - длительность звучания нот
```



songl // $<тактм><нота>

- $27,$27,$47,$27,$27,$47,$27,$2A,$23,$25,$87
- $28,$28,$28,$28,$28,$27,$27,$27,$27,$25,$25,$27,$45,$4A
- $27,$27,$47,$27,$27,$47,$27,$2A,$23,$25,$87
- $28,$28,$28,$28,$28,$27,$27,$27,$2A,$28,$27,$25,$83,$00

note // ноты $0..$B(0..11) и частоты нот

- 125,133,140,146,152,158,164,169,174,178,183,187,0,0,0,0
- // ЛЯ, ЛЯ#, СИ, ДО,ДО#,РЕ, РЕ#,МИ, ФА, ФА#,СОЛЬ,СОЛЬ#
- // 440,466,494,523,554,588,622,660,698,740,784, 831 Гц

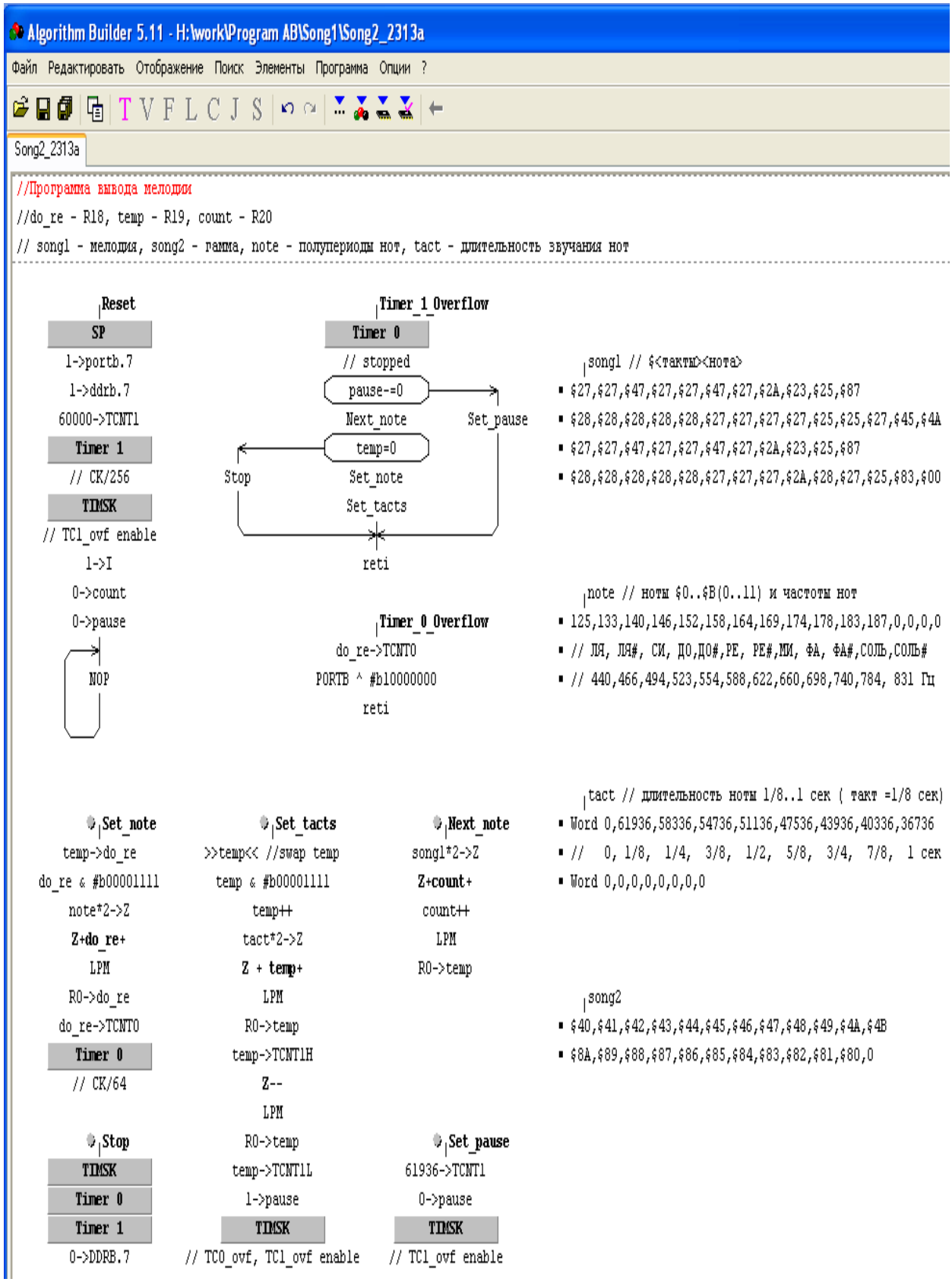
tact // длительность ноты 1/8..1 сек (такт =1/8 сек)

- Word 0,61936,58336,54736,51136,47536,43936,40336,36736
- // 0, 1/8, 1/4, 3/8, 1/2, 5/8, 3/4, 7/8, 1 сек

song2

- $40,$41,$42,$43,$44,$45,$46,$47,$48,$49,$4A,$4B
- $8A,$89,$88,$87,$86,$85,$84,$83,$82,$81,$80,0

Программа воспроизведения мелодии Song2_2313.alp с использованием макросов



ЛАБОРАТОРНАЯ РАБОТА №6
ИССЛЕДОВАНИЕ УНИВЕРСАЛЬНОГО АСИНХРОННОГО ПРИЁМОПЕРЕДАТЧИКА

ЦЕЛЬ РАБОТЫ

Получение сведений об особенностях программной и аппаратной реализации асинхронного обмена в AVR-микроконтроллерах.

Введение

Одной из важнейших составляющих работы современных вычислительных средств автоматики является обмен данными между устройствами. Зачастую при высокой скорости обработки данных производительность системы в целом оказывается недостаточной именно из-за ограничений, накладываемых возможностями в области обмена информацией между компонентами. В современных AVR-микроконтроллерах предусмотрено наличие встроенного универсального асинхронного приёмопередатчика (UART), обеспечивающего реализацию дуплексного асинхронного обмена с внешними устройствами. При построении систем с использованием AVR-микроконтроллеров следует учитывать следующие особенности UART, входящих в их состав:

- генерация произвольных значений скорости;
- высокая скорость при низких тактовых частотах;
- 8 или 9 бит данных;
- фильтрация шума;
- определение переполнения;
- детектирование ошибки кадра;
- определение неверного стартового бита;
- три раздельных прерывания - завершение передачи, очистка регистра передачи и завершение приема.

При программировании UART в простейшем случае решаются три задачи:

- инициализация UART (задание режимов работы);
- организация обмена с программным управлением по состоянию флагов;
- организация обмена данными с использованием прерываний.

Рассмотрим механизм решения этих задач:

Инициализация UART

Инициализация UART неизбежно предшествует процессу обмена. В противном случае результаты обмена информацией могут оказаться неадекватными, либо обмен окажется невозможен в принципе. В простейшем случае инициализация UART, входящего в состав AVR-микроконтроллера, включает следующие этапы:

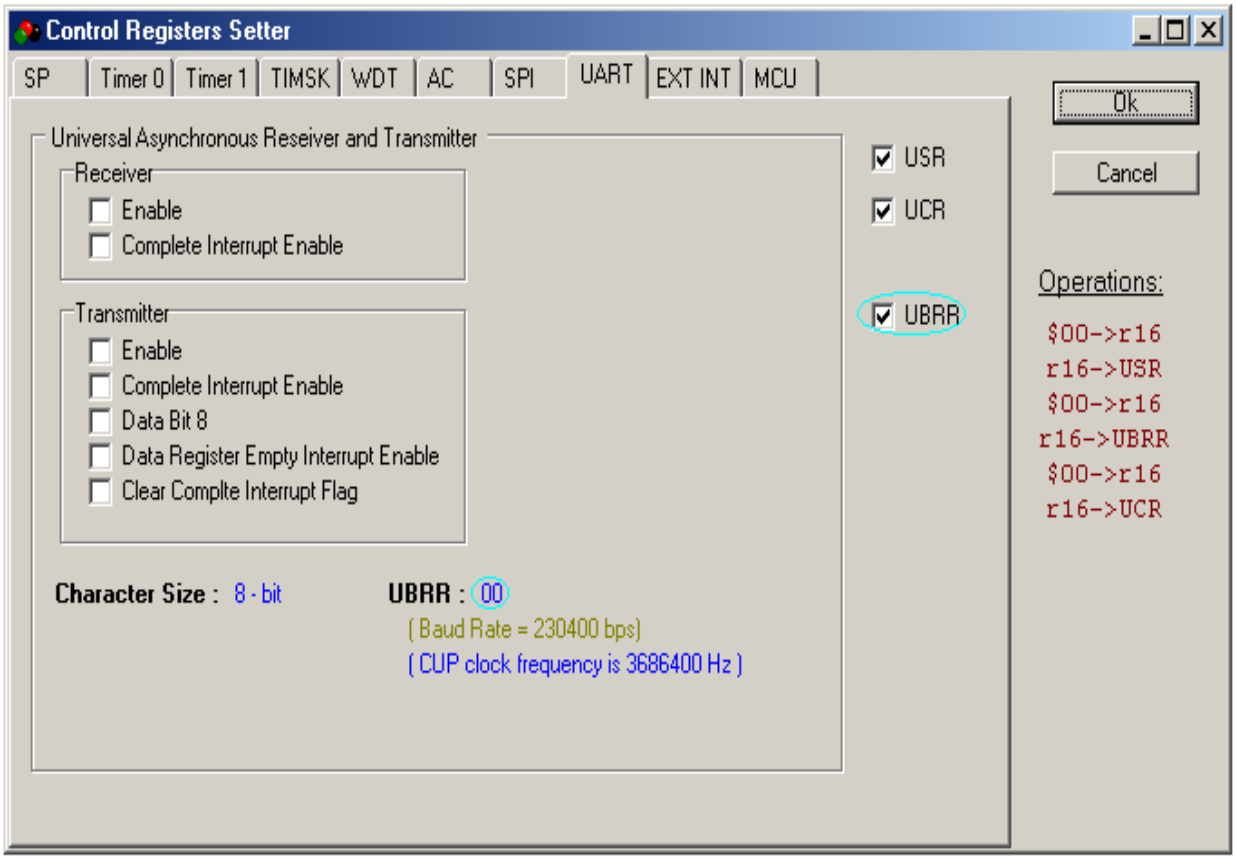
1. Установка скорости обмена. Поскольку при асинхронном обмене отсутствует отдельная линия синхронизации, для обеспечения корректности результатов приёма/передачи информации временные параметры сигналов должны быть строго согласованы по времени. Для поддержания стабильной скорости обмена в UART AVR-микроконтроллеров используется специальный генератор скорости передачи – делитель частоты, который генерирует скорости в соответствии с нижеприведенным выражением:

$$BAUD = \frac{F_{CK}}{16 \cdot (UBRR + 1)},$$

где
BAUD – скорость передачи;
F_{CK} – частота тактового генератора процессора;
UBRR - содержимое регистра скорости передачи UART.

В случае использования пакета “Algorithm Builder” установка скорости обмена осуществляется следующим образом:

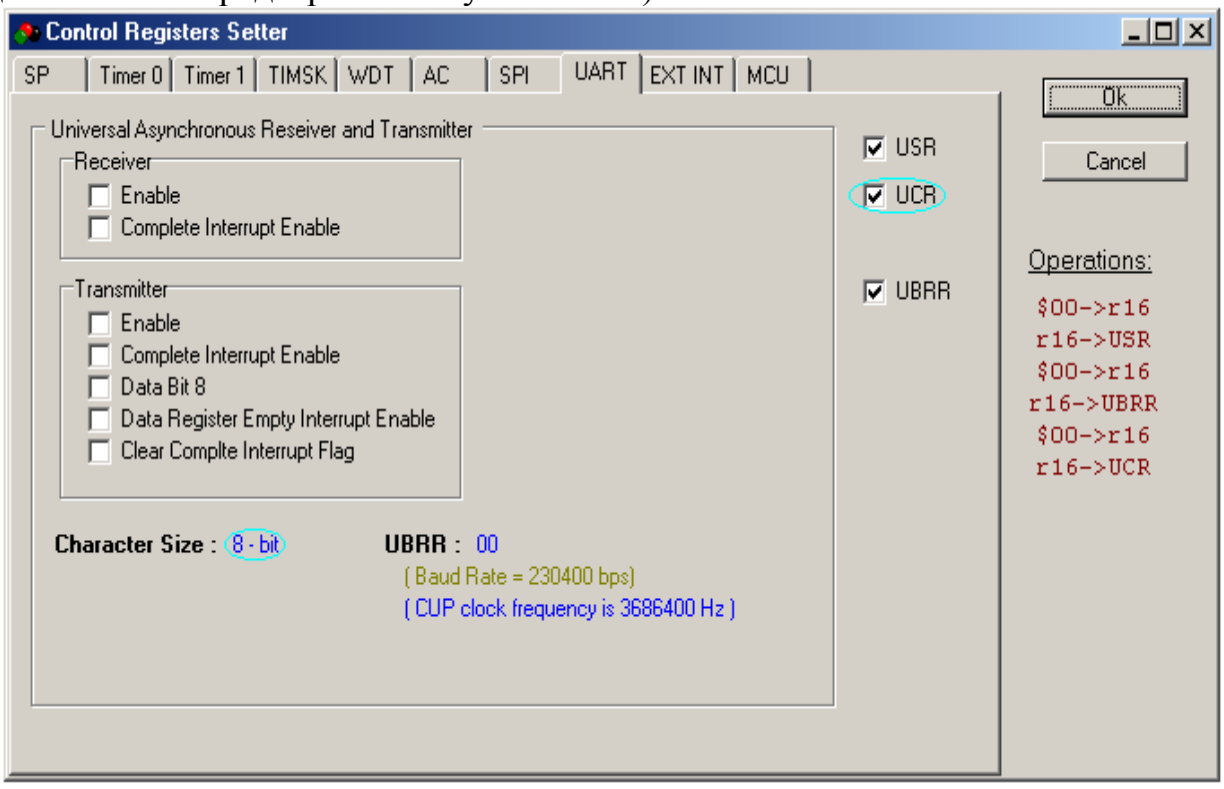
1) вставим в алгоритм элемент “Setter” (комбинация клавиш “Alt+S”);
2) в открывшемся окне выберем вкладку “UART”;
3) в соответствующей графе введём требуемое значение UBRR (флаг UBRR должен быть установлен):



При этом ниже будет отображена скорость передачи (BAUD). Целесообразно при установке скорости передачи пользоваться следующей таблицей:

Baud Rate	3.6864 MHz	Error, %	4 MHz	Error, %	7.3728 MHz	Error, %	8 MHz	Error, %
2400	UBRR=95	0.0	UBRR=103	0.2	UBRR=191	0.0	UBRR=207	0.2
4800	UBRR=47	0.0	UBRR=51	0.2	UBRR=95	0.0	UBRR=103	0.2
9600	UBRR=23	0.0	UBRR=25	0.2	UBRR=47	0.0	UBRR=51	0.2
14400	UBRR=15	0.0	UBRR=*16*	*2.1*	UBRR=31	0.0	UBRR=34	0.8
19200	UBRR=11	0.0	UBRR=12	0.2	UBRR=23	0.0	UBRR=25	0.2
28800	UBRR=7	0.0	UBRR=8	3.7	UBRR=15	0.0	UBRR=16	2.1
38400	UBRR=5	0.0	UBRR=6	7.5	UBRR=11	0.0	UBRR=12	0.2
57600	UBRR=3	0.0	UBRR=3	7.8	UBRR=7	0.0	UBRR=8	3.7
76800	UBRR=2	0.0	UBRR=2	7.8	UBRR=5	0.0	UBRR=6	7.5
115200	UBRR=1	0.0	UBRR=1	7.8	UBRR=3	0.0	UBRR=3	7.8

2. Установка формата посылки. Эта задача также решается с помощью уже введенного элемента “Setter”. Активизировав редактор двойным щелчком мыши, в соответствующей графе установим необходимый формат (флаг UCR должен быть предварительно установлен):



3. Управление приёмопередатчиком. Выполнение данной операции обеспечивается установкой управляющего регистра UCR и анализом флагов регистра состояния USR.

$0B ($2B)	USR	RXC	TXC	UDRE	FE	OR	-	-	-
$0A ($2A)	UCR	RXCIE	TXCIE	UDRIE	RXEN	TXEN	-	-	-

Состояние приемопередатчика определяется флагами: окончания приема (RxC), окончания передачи всех данных (TxC) , освобождения буфера передатчика (UDRE). Эти флаги могут вызывать прерывания при соответствующих установках регистра USR.

Разряд UCR	7	6	5	4	3	2	1	0
Название	RXCIE	TXCIE	UDRIE	RXEN	TXEN	0	0	0
При сбросе	0	0	0	0	0	0	0	0

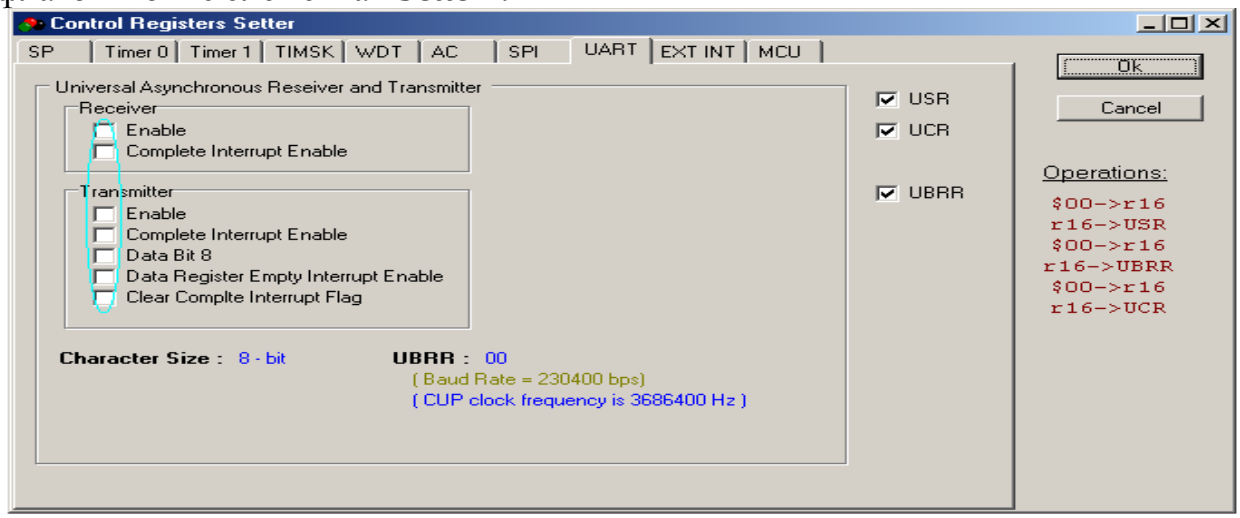
При сбросе все биты устанавливаются в 0 и приемопередатчик отключается. В секции инициализации программы его необходимо настроить. Назначение разрядов UCR.0.. UCR.7 описано ниже:

Таблица. Назначение разрядов управляющего регистра UCR

Разряд	**Название бита**	**Выполняемые функции**
UCR.0	0	
UCR.1	0	
UCR.2	0	
UCR.3	TXEN (Transmitter Enable)	Разрешение передачи UART, если установлен в "1".
UCR.4	RXEN (Receiver Enable)	Разрешение приема UART, если установлен в "1".
UCR.5	UDRIE* (UDR Empty Int. Enable)	Разрешение прерывание после очистки буфера данных передатчика, если установлен в "1".
UCR.6	TXCIE* (TX Complete Int. Enable)	Разрешение прерывание после окончания передачи всех данных передатчика, если установлен в "1".
UCR.7	RXCIE* (RX Complete Int. Enable)	Разрешение прерывание после окончания приема символа, если установлен в "1".

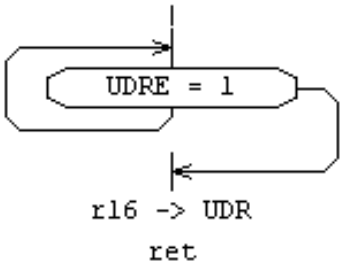
* Биты разрешения прерываний действуют при условии, что выполнено глобальное разрешение прерываний - SREG.7=I=1.

Аналогичные функции реализуются установкой соответствующих флагов в окне элемента “Setter”:

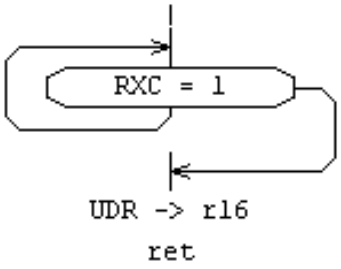


Организация обмена с программным управлением по состоянию флагов

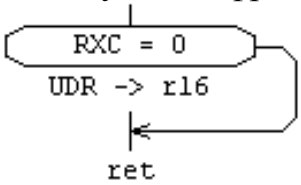
Поскольку UART реализует дуплексный режим обмена данными, возникает необходимость строгого согласования передачи и приёма по времени. Если микропроцессор использует последовательный канал для коротких однобайтных сообщений, то передавать и принимать данные можно, просто опрашивая флаги готовности передатчика и приемника UART. При передаче байта в последовательный канал все, что должен сделать процессор - дождаться установки флага готовности передатчика UDRE и записать затем передаваемый символ в регистр данных передатчика:



Аналогично, во время приема данных регистр данных приемника UART необходимо считывать лишь тогда, когда установлен флаг готовности RXC:



Вышеприведенный фрагмент обладает существенным недостатком: в том случае, если ожидаемые данные так и не поступят на вход UART, микроконтроллер «зависнет» (будет производиться бесконечный цикл опроса флага RXC). Этого недостатка лишён следующий фрагмент:



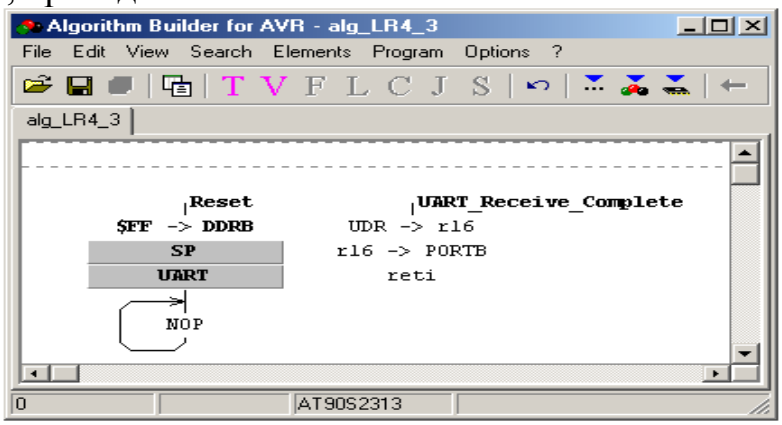
В этой подпрограмме вместо бесконечно долгого ожидания готовности приемника будет произведен лишь один опрос флага RXC. В случае, если он установлен, байт из регистра данных приемника будет считан, однако в любом случае подпрограмма будет завершена.

Проанализировав приведенные фрагменты, легко заметить, что и в случае передачи, и в случае приёма данных используется регистр UDR. Физически он представляет собой два отдельных регистра, доступ к которым происходит по одному адресу. При записи происходит запись в регистр передатчика, при чтении - читается регистр приемника.

Организация обмена с использованием прерываний

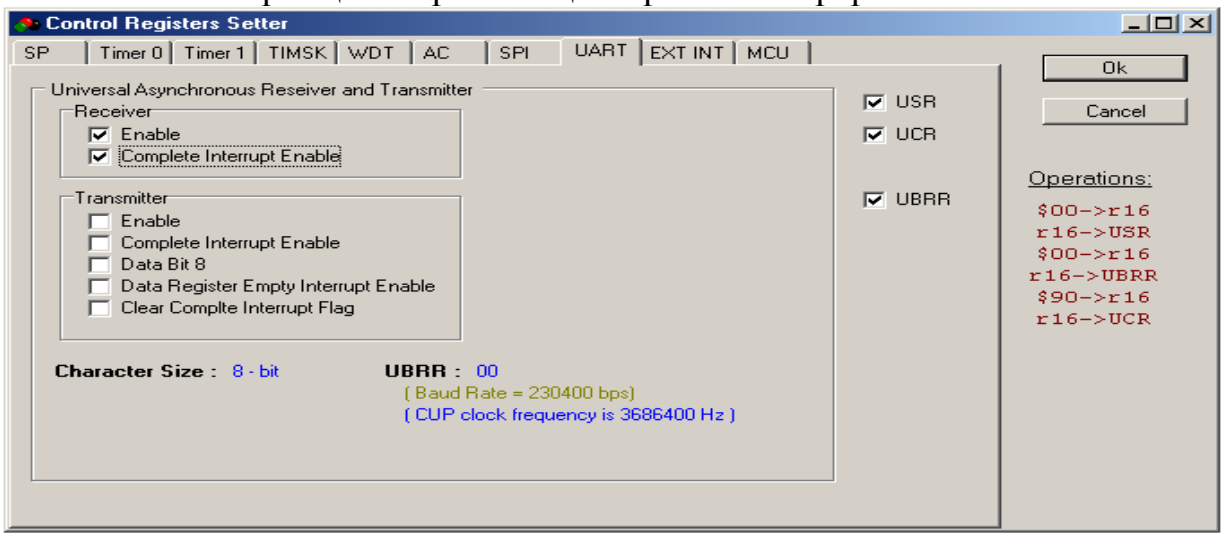
Рассмотрим реализацию приёма данных по прерыванию с использованием пакета “Algorithm Builder”. Реализация такого обмена возможна в том случае, если установлен бит RXCIE в регистре UCR. Тогда, при условии, что установлен бит глобального разрешения прерываний, установка бита RXC (флаг завершения приема UART, устанавливается в "1", когда принимаемый символ поступил из сдвигающего регистра приемника в регистр UDR, и обнуляется при чтении UDR) в регистре USR приводит к выполнению прерывания по окончании приема. Поскольку в случае, если после завершения приёма данных RXC сохранит единичное значение, при выходе из прерывания оно будет вызвано снова, при использовании приема данных по прерыванию обработчик прерывания должен читать регистр UDR для сброса RXC.

Фрагмент, отражающий принцип организации приёма с использованием прерываний, приведен ниже:



Обратим внимание на то, что элементу настройки UART предшествуют поле инициализации порта B и указатель позиции стека. Попутно заметим, что поле инициализации порта B имеет форму макрооператора (клавиша F2).

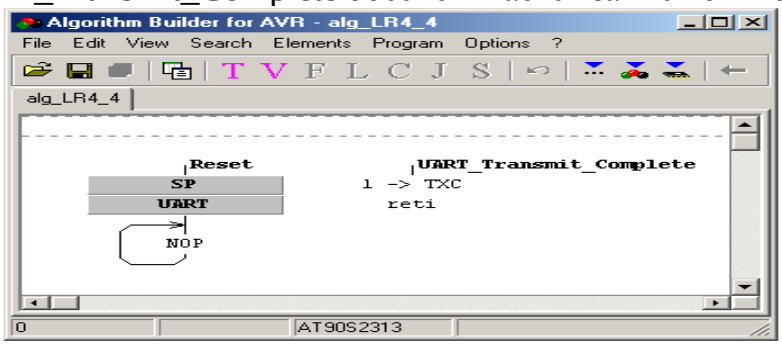
Флаги в окне настройки UART устанавливаются в соответствии с вышеизложенным принципом реализации приёма по прерыванию:



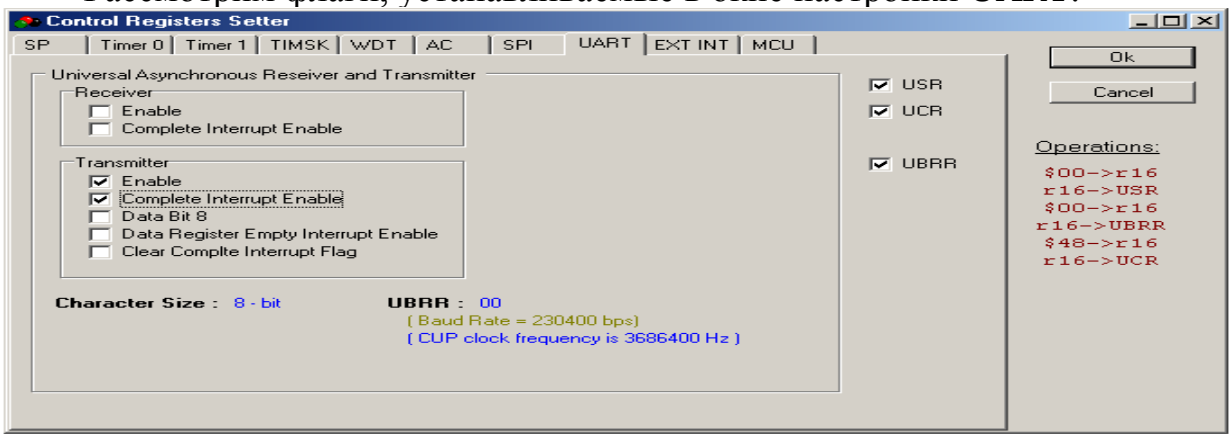
Флагу *Enable* поля приёмника соответствует бит RXEN регистра UCR, активизирующий приёмник UART. Флагу *Complete Interrupt Enable* соответствует бит RXCIE регистра UCR, разрешающий прерывание по завершении приёма. Это прерывание приводит к выполнению подпрограммы UART_Receive_Complete. В ходе её выполнения производится чтение регистра UDR (как уже упоминалось выше, это необходимо для сброса бита RXC регистра USR). Затем содержимое UDR через регистр r16 выводится в порт B. После этого следует команда выход из прерывания *reti*.

Аналогично реализуется передача данных по прерыванию. Для этого должен быть установлен бит TXCIE регистра UCR. В этом случае установка бита TXC (флаг завершения передачи UART) приведёт к выполнению прерывания. При этом будет выполнен обработчик прерывания UART_Transmit_Complete, производящий сброс бита TXC статусного регистра USR с последующим выходом из прерывания (команда *reti*). Фрагмент, реализующий вышеописанные операции, приведен ниже.

Обратим внимание на то, что, как и в случае приёма данных по прерыванию, элементу настройки UART предшествует указатель позиции стека. Однако, в отличие от RXC, сброс которого производился чтением UDR, сброс TXC в подпрограмме UART_Transmit_Complete обеспечивается записью в него 1.



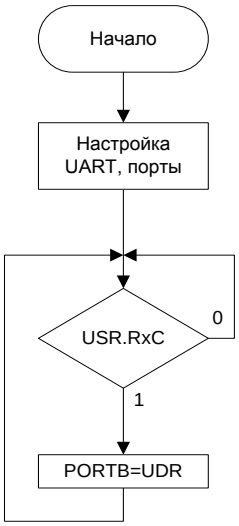
Рассмотрим флаги, устанавливаемые в окне настройки UART:



Флагу *Enable* поля передатчика соответствует бит TXEN регистра UCR, активизирующий передатчик UART. Флагу *Complete Interrupt Enable* соответствует бит TXCIE регистра UCR, разрешающий прерывание по завершении передачи. Следует иметь в виду, что выполнение подпрограммы UART_Transmit_Complete приводит к сбросу бита глобального разрешения прерываний I.

Таким образом, во время обработки прерывания, вызванного установкой бита TXC, все остальные прерывания запрещены. После выполнения команды выхода из прерывания *reti* значение бита глобального разрешения прерываний восстанавливается.

Задание 1. Программирование ввода данных



Алгоритм работы МК

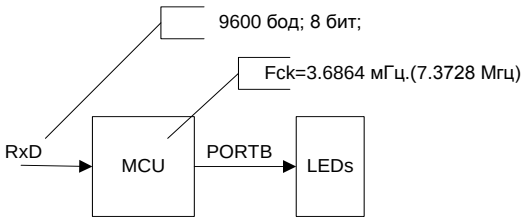
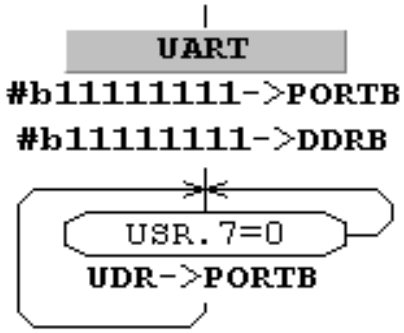
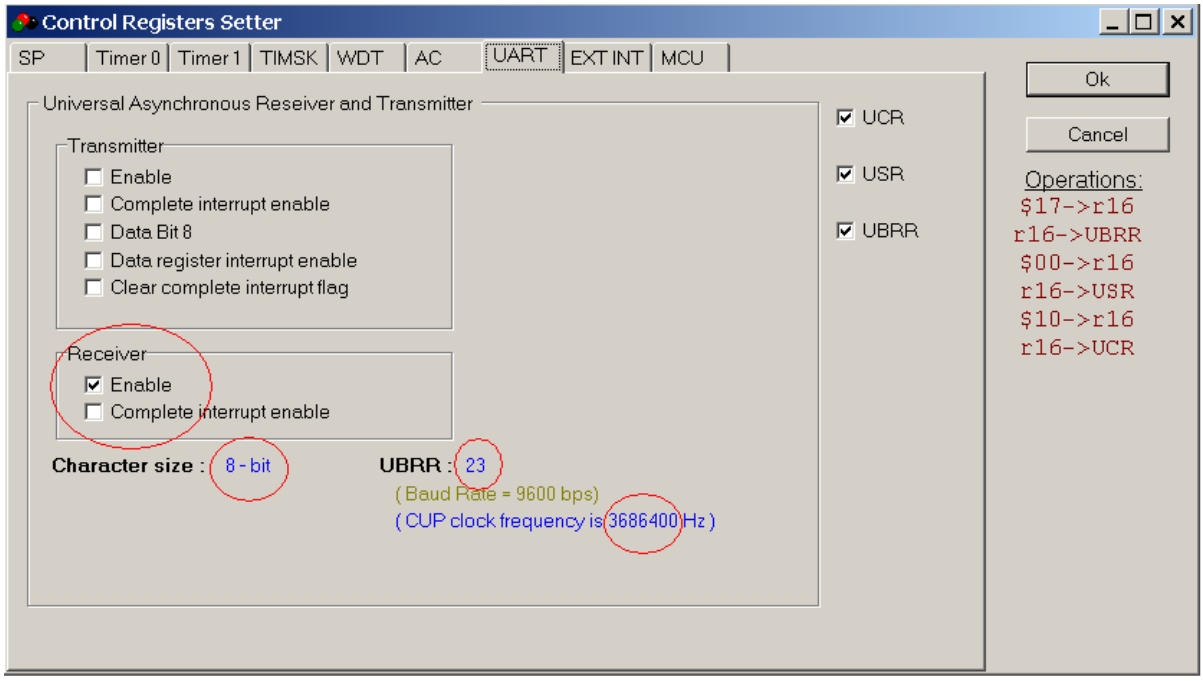


Схема эксперимента



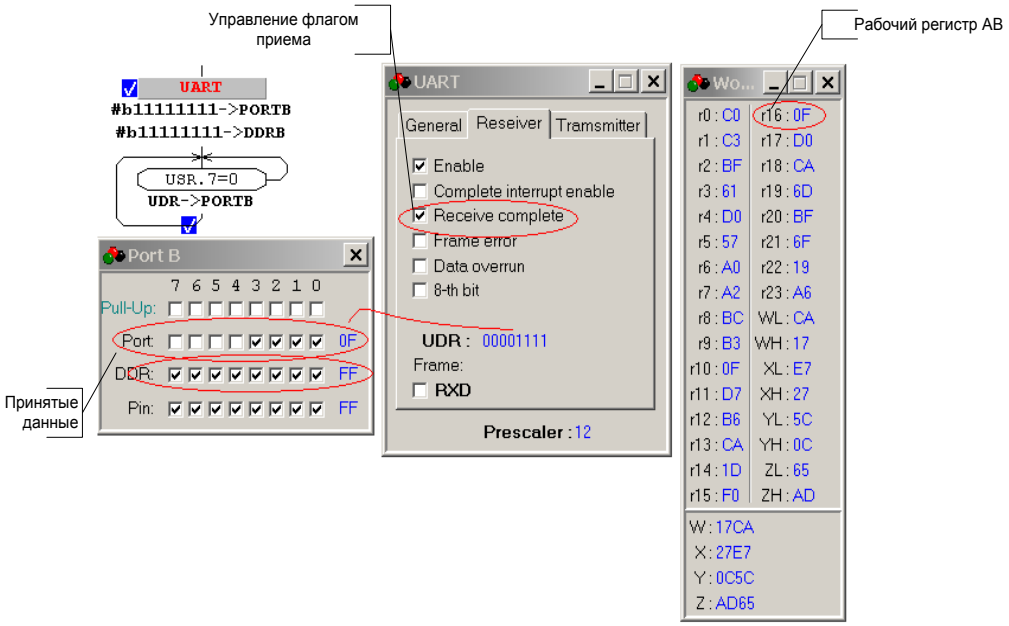
Программа в Algorithm Builder



Настройка приемника UART для STK 500

При использовании других аппаратных средств, кроме STK500, необходимо изменить значение тактовой частоты и коэффициент деления в UBRR.

Отладка программы в симуляторе Algorithm Builder (AB)

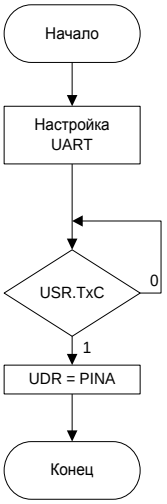


Рабочее окно симулятора Algorithm Builder

Последовательность действий при отладке:

1. Запуск в пошаговом режиме (12-15 шагов)
2. Установка значения принятого кода в UDR
3. Установка флага окончания приема Recieve Complete (RxC)
4. Выполнить 5-7 шагов, наблюдая изменение состояния порта B и переход к ожиданию поступления очередного кода. Пересылка из UDR в PORTB выполняется через рабочий регистр R16
5. Повторить 3-5 раз пп.2-4

`Задание 2. Программирование вывода данных`



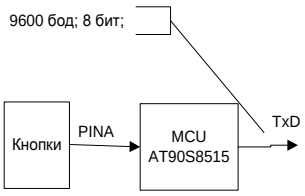


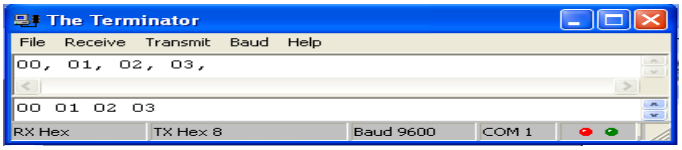
Схема эксперимента

Организовать программу циклического вывода с периодом 20 мс.

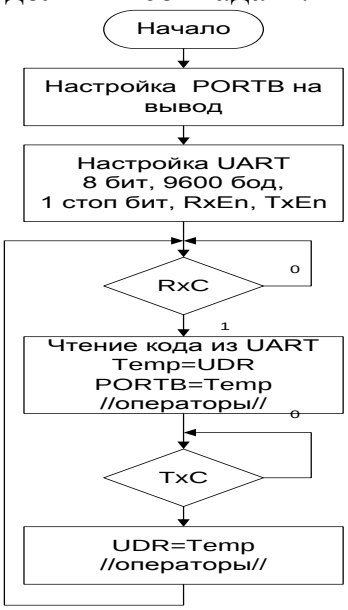
Задание 3. Программирование последовательного обмена микроконтроллера и ПК через COM- порт

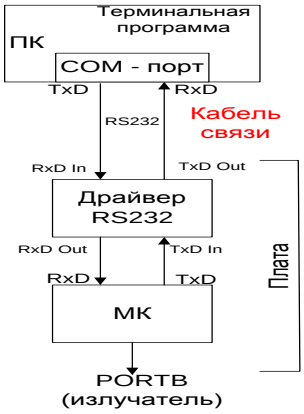
Схема алгоритма включает программный опрос флага готовности приемника UART, ожидание приема байта (RxC=1). Принятый байт выводится в порт B и передается обратно компьютеру после освобождения передатчика (TxC=1) – режим эхо.

Для передачи последовательности байт данных используется терминальная программа.



В нижнем окне набираются байты в выбранном режиме для передачи, а в верхнем окне отображаются принятые байты. Эти последовательности в режиме «эхо» должны совпадать.





Задание 4. Включить в программу генератор тонального сигнала, период которого задается принятым кодом.

С помощью терминальной программы создать последовательность кодов для управления воспроизведения простого фрагмента мелодии.

Задание 5. Разработать и отладить программу приема произвольной последовательности символов и при обнаружении заданной 3-4 символьной последовательности включать на короткое (100-200мс) время звуковой сигнал и отправлять подтверждение – символ «#».

Задание 6. Построить программу приема и передачи данных с использованием прерываний.

Содержание отчета

1. Формат регистров, связанных с UART, назначение бит.
2. Исследуемые программы и комментарии к ним.
3. Схемы экспериментов.
4. Тестовые последовательности для проверки обмена.
5. Выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №7
ИССЛЕДОВАНИЕ ПОСЛЕДОВАТЕЛЬНОГО ПЕРИФЕРИЙНОГО ИНТЕРФЕЙСА SPI

ЦЕЛЬ РАБОТЫ

Получение сведений о возможностях, связанных с использованием интерфейса SPI в AVR-микроконтроллерах.

Введение

Последовательный периферийный интерфейс SPI (Serial Peripheral Interface) реализован в большинстве современных AVR-микроконтроллеров. Он представляет собой четырёхпроводный интерфейс, позволяющий реализовать синхронный полнодуплексный обмен данными. К достоинствам интерфейса относят высокую скорость передачи и возможность выделения отдельной линии синхронизации. Спектр областей применения SPI достаточно широк:

- осуществление обмена данными между микроконтроллером и различными периферийными устройствами (цифровые потенциометры, ЦАП, АЦП, Flash-ПЗУ и др.);
- осуществление обмена между несколькими AVR-микроконтроллерами (использование SPI в качестве высокоскоростного канала связи);
- программирование микроконтроллера (т. н. режим последовательного программирования).

При обмене данными по интерфейсу SPI AVR-микроконтроллер может работать как ведущий (режим «Master») либо как ведомый (режим «Slave»). Ведущая роль Master-устройства состоит в том, что оно управляет темпом обмена, выдавая импульсы синхронизации, и является инициатором обмена. При этом пользователь может задавать скорость передачи (семь программируемых значений) и формат передачи (от младшего разряда к старшему или наоборот). Простота интерфейса позволяет использовать в качестве Slave-устройства одну микросхему сдвигового регистра.

Дополнительной возможностью подсистемы SPI является «пробуждение» микроконтроллера из режима Idle (режим холостого хода, при переходе в который ЦПУ останавливается). Сигналом к выходу из режима Idle служит поступление данных.

При программировании SPI в простейшем случае решаются три задачи:

- инициализация SPI (задание режимов работы);
- организация обмена с программным управлением по состоянию флагов;
- организация обмена данными с использованием прерываний.

Рассмотрим механизм решения этих задач:

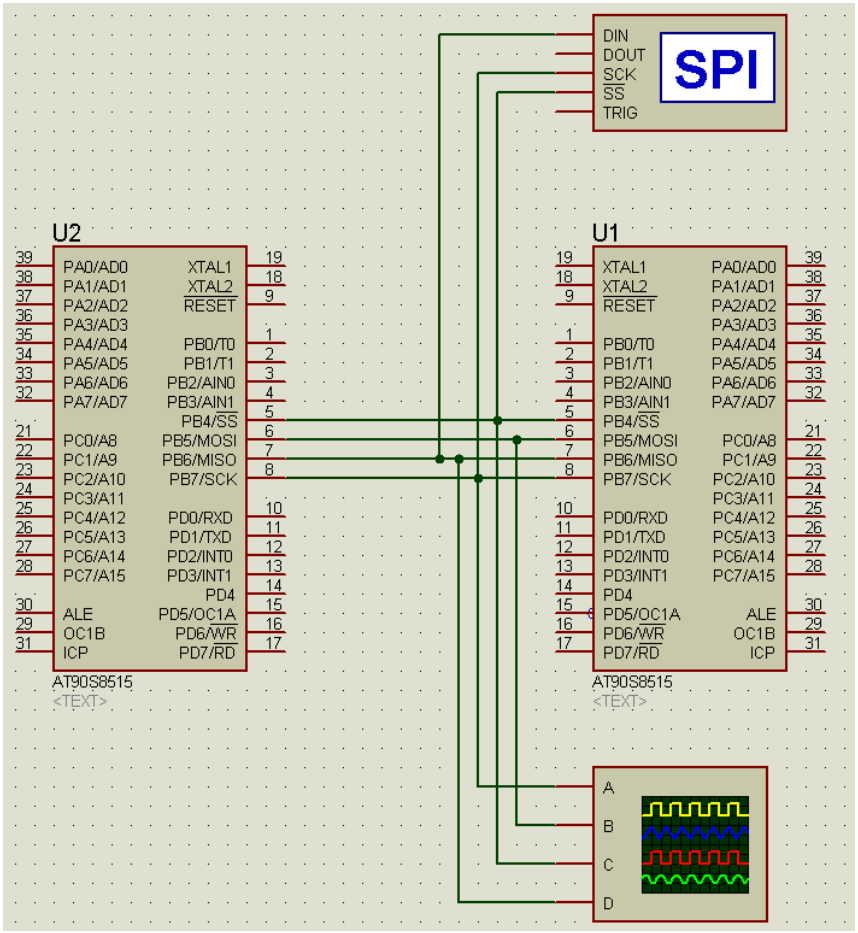


Схема связи микроконтроллеров по интерфейсу SPI (U1 – Master, U2- Slave)

Инициализация SPI

Инициализация SPI неизбежно предшествует процессу обмена. В противном случае результаты обмена информацией могут оказаться неадекватными, либо обмен окажется невозможен в принципе. В простейшем случае инициализация SPI, входящего в состав AVR-микроконтроллера, включает следующие этапы:

1. Включение SPI. Перед выполнением обмена необходимо, прежде всего, разрешить работу модуля SPI. Для этого следует установить в "1" разряд SPE регистра SPCR (см. рис.). Как уже упоминалось выше, AVR-микроконтроллер может работать как ведущий (режим «Master») либо как ведомый (режим «Slave»). Режим работы определяется состоянием разряда MSTR регистра SPCR: если разряд установлен в "1", микроконтроллер работает в режиме «Master», если сброшен в "0" – в режиме «Slave».

Разряд	7	6	5	4	3	2	1	0
Название	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
Начальное значение	0	0	0	0	0	0	0	0

Формат регистра SPCR

2. Установка скорости обмена. Поскольку при синхронном обмене имеется отдельная линия синхронизации (SCK), для обеспечения корректности результатов приёма/передачи информации используются импульсы синхронизации, выдаваемые Master-устройством и передаваемые по этой линии на соответствующий вход Slave-устройства. Таким образом, Master-устройство задаёт темп обмена. Для установки скорости передачи используются разряды SPR1 и SPR0 регистра SPCR. Таблица, отражающая соответствие между состоянием этих разрядов и устанавливаемым коэффициентом деления частоты F_{CK} тактового генератора процессора, приведена ниже:

SPR1	**SPR0**	**частота FCK**
0	0	F_{CK} /4
0	1	F_{CK} /16
1	0	F_{CK} /64
1	1	F_{CK} /128

Дополнительное управление скоростью осуществляется с помощью разряда SPI2X регистра SPSR (см рис.): при установке этого разряда в "1" частота следования синхроимпульсов Master-устройства удваивается.

Разряд	7	6	5	4	3	2	1	0
Название	SPIF	WCOL	–	–	–	–	–	SPI2X
Начальное значение	0	0	0	0	0	0	0	0

Формат регистра SPSR

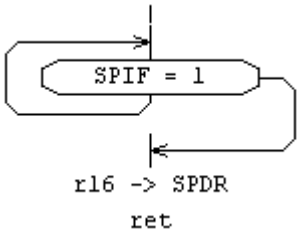
Примечание: разряды с 5-го по 1-й регистра SPSR зарезервированы и читаются как "0".

3. Установка режима передачи данных. Спецификация интерфейса SPI предусматривает 4 режима передачи данных. Эти режимы различаются соответствием между фазой (моментом считывания сигнала) тактового сигнала SCK, его полярностью и передаваемыми данными. Всего существует 4 таких комбинации, определяемых состоянием разрядов CPHA и CPOL регистра SPCR. Если CPOL сброшен в "0", генерируются импульсы положительной полярности, при отсутствии импульсов на выводе присутствует низкий уровень; если CPOL установлен в "1", генерируются импульсы отрицательной полярности, при отсутствии импульсов на выводе присутствует высокий уровень. Если CPHA сброшен в "0", обработка данных производится по переднему фронту импульсов SCK; если CPHA установлен в "1", обработка данных производится по заднему фронту импульсов SCK.

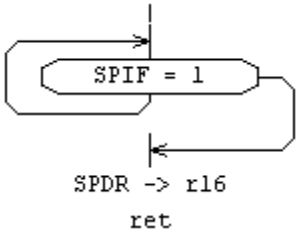
Кроме того, интерфейс SPI позволяет определить порядок передачи разрядов данных. Для этой цели служит разряд DORD регистра SPCR: если разряд установлен в "1", первым передаётся младший разряд байта, в противном случае – старший разряд.

Организация обмена с программным управлением по состоянию флагов

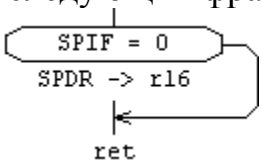
Поскольку SPI реализует полнодуплексный режим обмена данными, возникает необходимость строгого согласования передачи и приёма по времени. В противном случае часть передаваемых байтов может быть потеряна. Обмен данными в модуле SPI организован таким образом, что готовый для передачи байт данных не может быть записан в регистр данных SPI до окончания предыдущего цикла обмена. Таким образом, передавать и принимать данные можно, просто опрашивая флаг SPIF регистра SPSR (данный флаг устанавливается в "1" по окончании передачи очередного байта). При передаче байта в буфер все, что должен сделать процессор - дождаться установки флага SPIF и записать затем передаваемый символ в регистр данных передатчика:



Аналогично, во время приема данных регистр данных SPDR необходимо считывать лишь тогда, когда установлен флаг SPIF:



Вышеприведенный фрагмент обладает существенным недостатком: в том случае, если ожидаемые данные так и не поступят на вход SPDR, микроконтроллер «зависнет» (будет производиться бесконечный цикл опроса флага SPIF). Этого недостатка лишён следующий фрагмент:



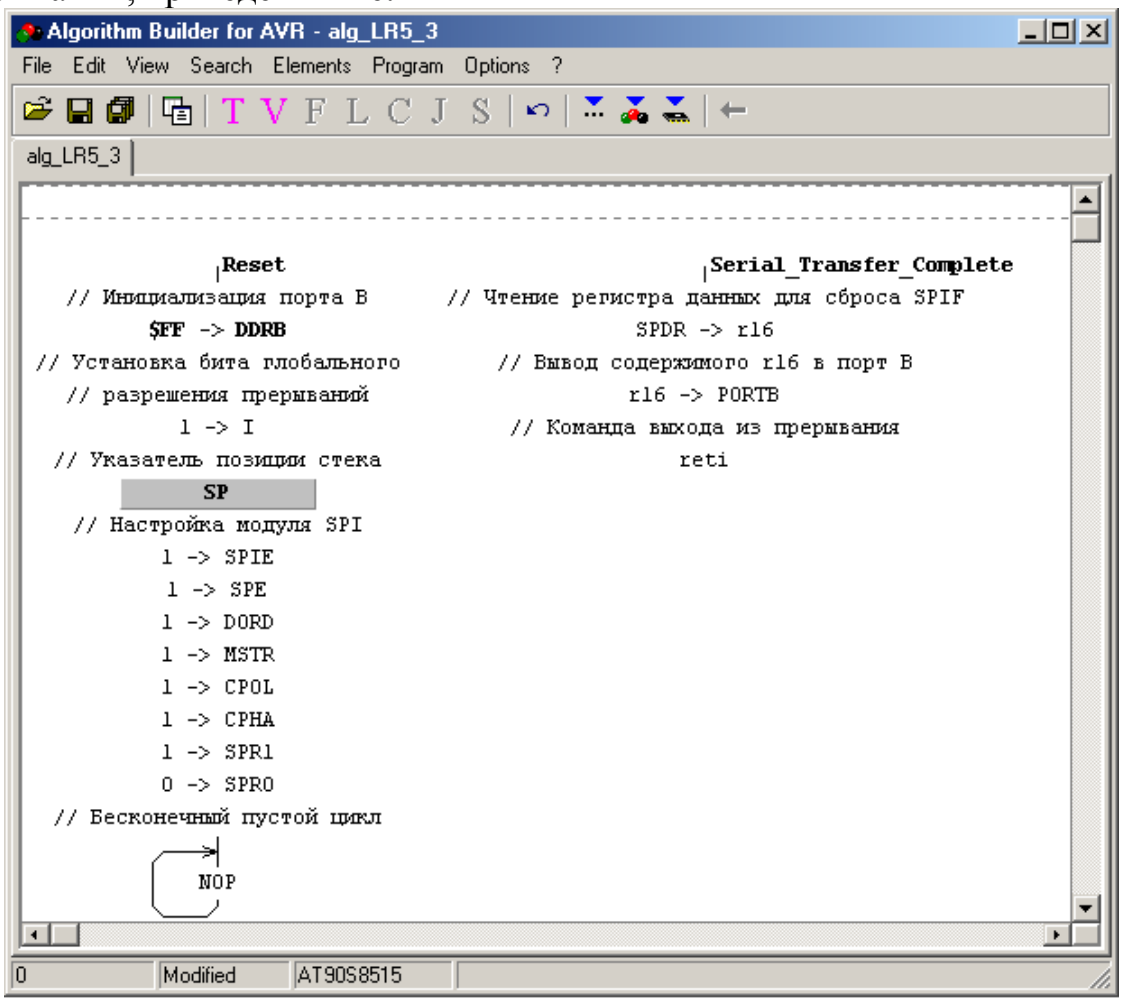
В этой подпрограмме вместо бесконечно долгого ожидания готовности приемника будет произведен лишь один опрос флага SPIF. В случае, если он установлен, байт из регистра данных приемника будет считан, однако в любом случае подпрограмма будет завершена.

Проанализировав приведенные фрагменты, легко заметить, что и в случае передачи, и в случае приёма данных используется регистр SPDR. Запись в этот регистр инициирует начало передачи, а при его чтении считывается содержимое буфера сдвигового регистра. Другими словами, регистр данных служит буфером между регистровым файлом микроконтроллера и сдвиговым регистром модуля SPI.

Организация обмена с использованием прерываний

Рассмотрим реализацию обмена данными по прерыванию в SPI с использованием пакета “Algorithm Builder”. Реализация такого обмена возможна в том случае, если установлен бит SPIE в регистре SPCR. Тогда, при условии, что установлен бит глобального разрешения прерываний, установка бита SPIF (флаг завершения цикла обмена, устанавливается в "1", когда последний разряд текущего байта поступил из регистра данных SPDR в сдвиговый регистр модуля SPI, и сбрасывается в "0" при чтении SPDR) в регистре SPSR приводит к выполнению прерывания по окончании передачи. Поскольку в случае, если после завершения обмена данными SPIF сохранит единичное значение, при выходе из прерывания оно будет вызвано снова, при использовании обмена данными по прерыванию обработчик прерывания должен читать регистр SPDR для сброса SPIF.

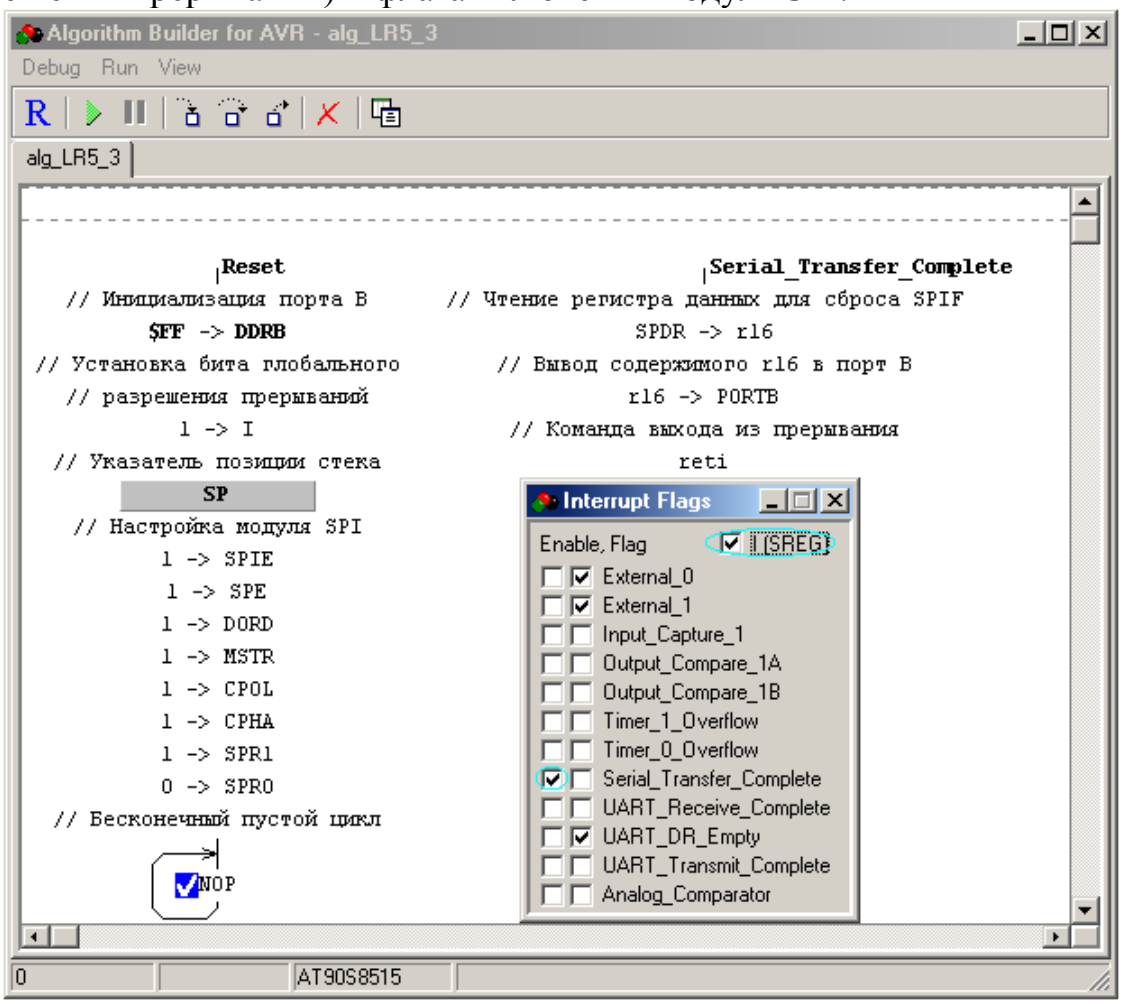
Фрагмент, отражающий принцип организации обмена с использованием прерываний, приведен ниже:



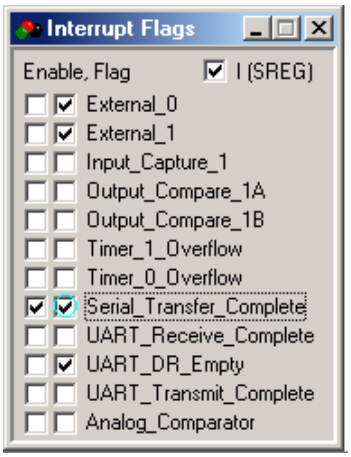
Обратим внимание на то, что вводу настроек SPI предшествуют поле инициализации порта B и указатель позиции стека. Попутно заметим, что поле инициализации порта B имеет форму макрооператора (клавиша F2).

Следует иметь в виду, что выполнение подпрограммы Serial_Transfer_Complete приводит к сбросу бита глобального разрешения прерываний I. Таким образом, во время обработки прерывания, вызванного установкой бита SPIF, все остальные прерывания запрещены. После выполнения команды выхода из прерывания *reti* значение бита глобального разрешения прерываний восстанавливается.

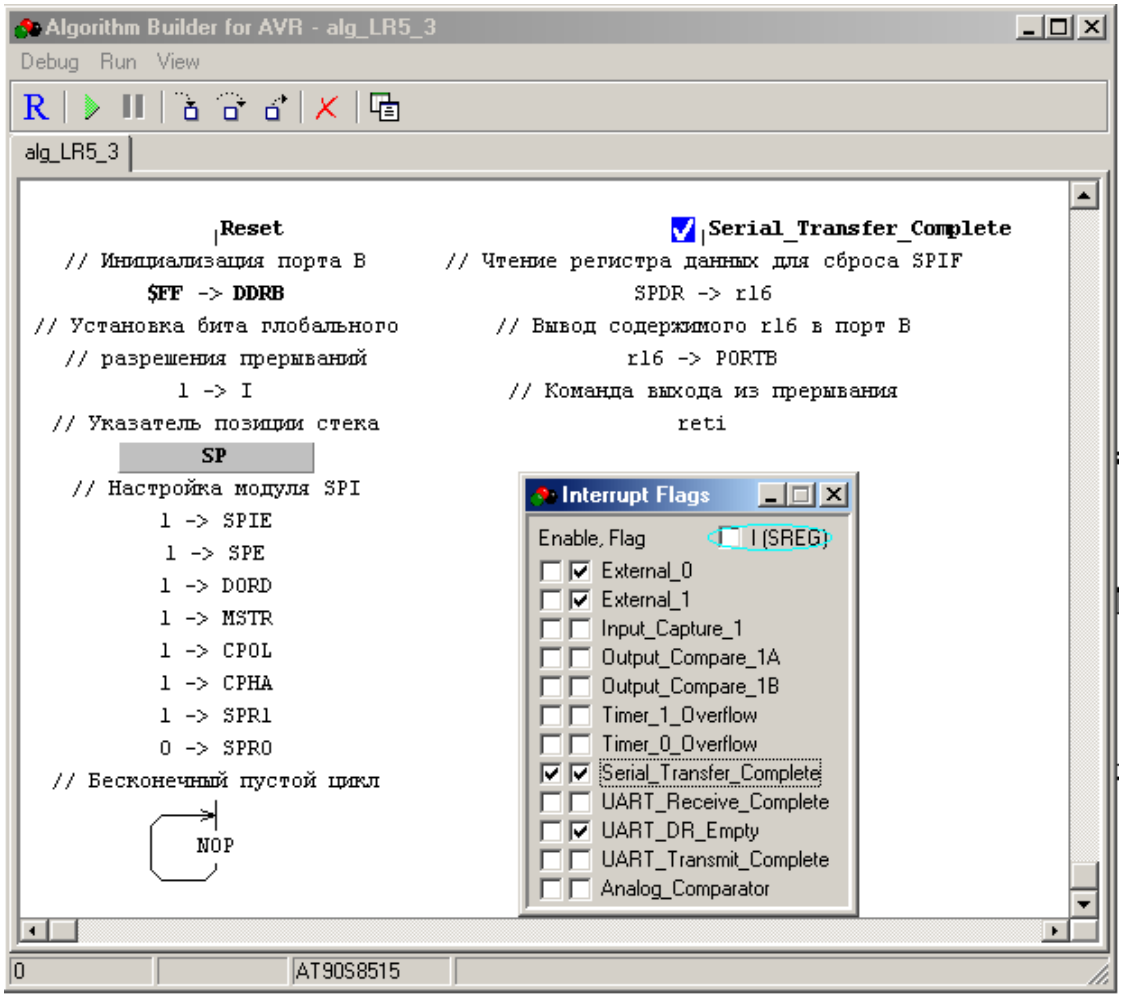
Предварительно выбрав тип используемого микроконтроллера (пункт "***Project options…***" меню "***Options***"), исследуем работу вышеприведенного фрагмента в симуляторе (клавиша "F9"). Для наблюдения за протеканием процессов откроем окно "Interrupt Flags" (пункт "***Interrupts***" меню "***View***"). Запустив алгоритм на непрерывное исполнение повторным нажатием клавиши "F9", остановим работу симулятора, нажав клавишу "F2". Обратим внимание на то, что метка ☑, означающая текущее положение программного счетчика, установлена внутри бесконечного пустого цикла. Также заметим, что в окне "Interrupt Flags" произошла установка флага I (бит глобального разрешения прерываний) и флага включения модуля SPI:



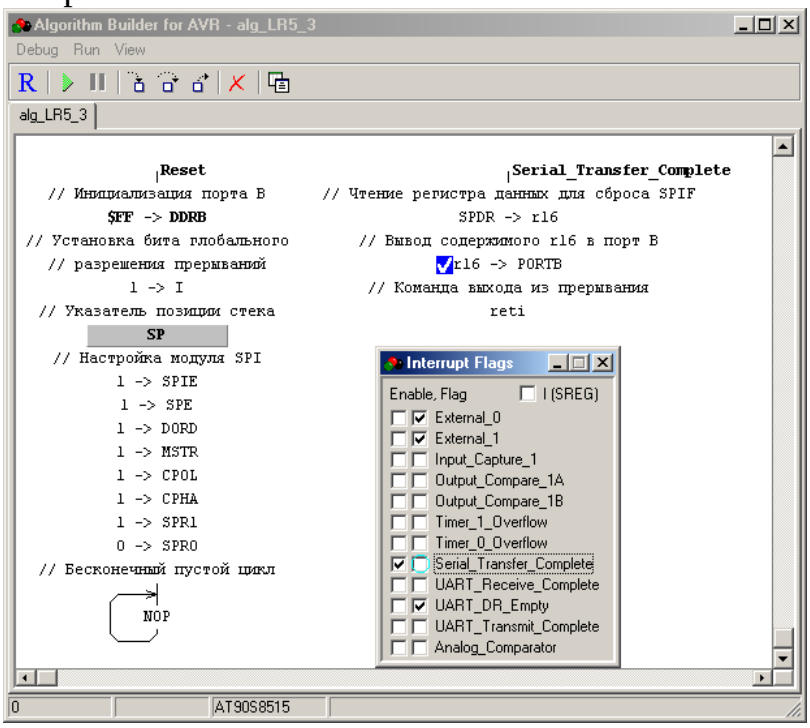
Далее, имитируем окончание передачи байта, вручную установив флаг Serial_Transfer_Complete, соответствующий биту SPIF регистра SPSR:



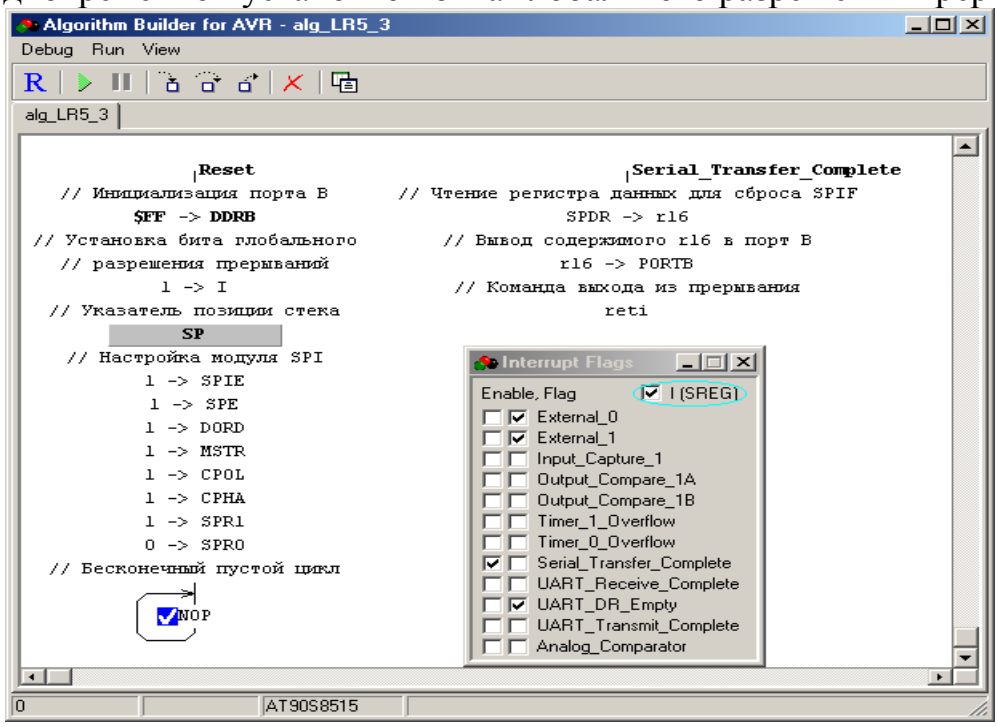
После этого, воспользовавшись возможностью пошагового исполнения алгоритма (клавиша “F7”), убедимся в том, что в следующем после установки флага Serial_Transfer_Complete шаге происходит переход к выполнению под-программы обработки прерывания. Также заметим, что, как и было указано ранее, при выполнении обработки прерывания сбрасывается бит глобального разрешения прерываний I:



Затем, пользуясь клавишей “F7” для пошагового исполнения алгоритма, убедимся, что при чтении регистра данных SPDR происходит сброс флага Serial_Transfer_Complete:



Дальнейшее пошаговое исполнение алгоритма после выполнения команды выхода из прерывания *reti* приводит к возврату в бесконечный пустой цикл с одновременной установкой бита глобального разрешения прерываний:



После этого ведущий микроконтроллер может начать передачу следующего байта.

ЛАБОРАТОРНАЯ РАБОТА №8
ИССЛЕДОВАНИЕ АНАЛОГОВОГО ИНТЕРФЕЙСА МИКРОКОНТРОЛЛЕРА

Цель работы: Изучение элементов управления аналоговым компаратором и аналого – цифровым преобразователем, построение и отладка программ с использованием аналогового интерфейса.

Программа работы:

1. Изучение назначения бит регистров, связанных с компаратором и АЦП
2. Построение и отладка программы с использованием компаратора
   - без использования прерывания
   - С использованием прерывания
3. Построение и отладка программы с использованием АЦП
   - без использования прерывания
   - С использованием прерывания

1. Аналоговый компаратор

Аналоговый компаратор сравнивает уровни на положительном выводе PB0(AIN0) и отрицательном выводе PB1(AIN1). При напряжении на положительном выводе PB0(AIN0) большем, чем напряжение на отрицательном выводе PB1(AIN1), выход аналогового компаратора ACO устанавливается в состояние 1. Выход компаратора может быть использован для управления входом захвата таймера/счетчика1. Кроме того, компаратор может формировать свой запрос прерывания. Пользователь может задать формирование запроса на прерывание по нарастающему или падающему фронту или по переключению. Блок-схема аналогового компаратора, со схемами обрамления, показана на рис. 1.

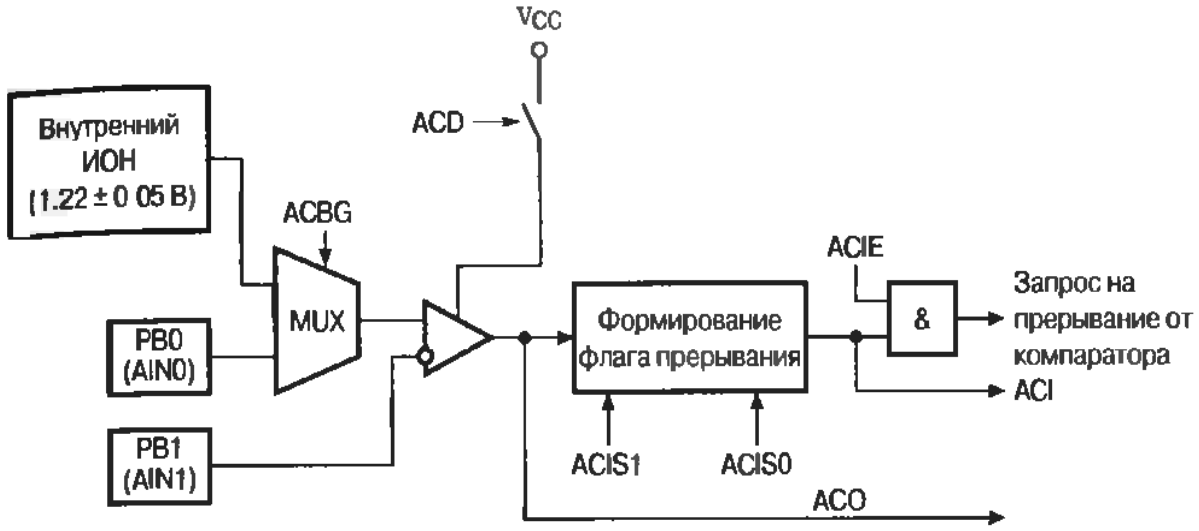


Рис. 1. Блок-схема аналогового компаратора

Регистр статуса и управления аналогового компаратора - ACSR
(The Analog Comparator Control and Status Register)

Биты	7	6	5	4	3	2	1	0	
$08 ($28)	**ACD**	**ACBG**	**ACO**	**ACI**	**ACIE**	**-**	**ACIS1**	**ACIS0**	**ACSR**
Чтение/Запись	R/W	R	R	R/W	R/W	R/W	R/W	R/W	
Начальное состояние	0	0	0	0	0	0	0	0	

- **Bit 7 - ACD: Analog Comparator Disable - Запрет аналогового компаратора.** При установленном в состояние 1 бите ACD аналоговый компаратор запрещен. Для выключения аналогового компаратора установку данного бита можно производить в любое время. Отключение аналогового компаратора позволяет снизить потребление в активном и Idle режимах. При изменении состояния бита ACD необходимо запрещать прерывание по аналоговому компаратору очисткой бита ACIE в регистре ACSR. В противном случае при изменении состояния бита ACD может произойти прерывание.
- **Bit 6 – ACBG: Analog Comparator Bandgap Select –** Подключение вместо AIN0 напряжения внутреннего источника опорного напряжения 1.22 В.
- **Bit 5 - ACO: Analog Comparator Output - Выход аналогового компаратора.** Бит ACO связан непосредственно с выходом компаратора.
- **Bit 4 - ACI: Analog Comparator Interrupt Flag - Флаг прерывания по аналоговому компаратору.** Данный бит устанавливается в состояние 1 в случае формирования компаратором прерывания, определяемого ACIS1 и ACIS0. Подпрограмма обработки прерывания по аналоговому компаратору будет выполняться при установленном бите ACIE и установленном бите глобального прерывания в регистре SREG. Бит ACI очищается аппаратно при выполнении соответствующего вектора обработки прерывания, Бит ACI можно очистить, также, записью во флаг логической 1.
- **Bit 3 - ACIE: Analog Comparator Interrupt Enable - Разрешение прерывания по аналоговому компаратору.** При установленном бите ACIE и установленном бите глобального прерывания регистра SREG активируется прерывание по аналоговому компаратору. При сброшенном бите ACIE прерывание запрещено.
- **Bit 2 - ACIC: Analog Comparator Input Capture enable - Разрешение входа захвата аналогового компаратора. Bits 1,0 - ACIS1, ACIS0: Analog Comparator Interrupt Mode Select - Выбор режима прерывания по аналоговому компаратору.** Эти биты определяют характер события компаратора, при котором запускается прерывание по аналоговому компаратору. Варианты установок показаны в таблице 1.

Таблица 1. Установки битов ACIS1/ACIS0

ACIS1	ACIS0	Режим прерывания
0	0	Прерывание по переключению выхода компаратора
0	1	Зарезервировано
1	0	Прерывание по падающему фронту на выходе компаратора
1	1	Прерывание по нарастающему фронту на выходе компаратора

Примечание: При изменении состояния битов ACIS1/ACIS0 прерывание по аналоговому компаратору должно быть запрещено очисткой бита разрешения прерывания в регистре ACSR. В противном случае при изменении состояния битов может произойти прерывание.

2. Аналого-цифровой преобразователь - (Analog to Digital Converter)

Основные характеристики:

- Разрешение 10 разрядов
- Точность ±1/2 LSB
- Время преобразования 70...280 мс
- 8 мультиплексируемых каналов входа
- Режимы циклического и однократного преобразования
- Прерывание по завершению ADC преобразования
- Устройство подавления шумов Sleep режима

Микроконтроллеры AVR оснащены 10-разрядным ADC последовательного приближения. ADC подсоединен к 8-канальному аналоговому мультиплексору, позволяющему использовать один из выводов порта в качестве входа ADC. ADC содержит усилитель выборки/хранения, удерживающий напряжение входа ADC во время преобразования на неизменном уровне.

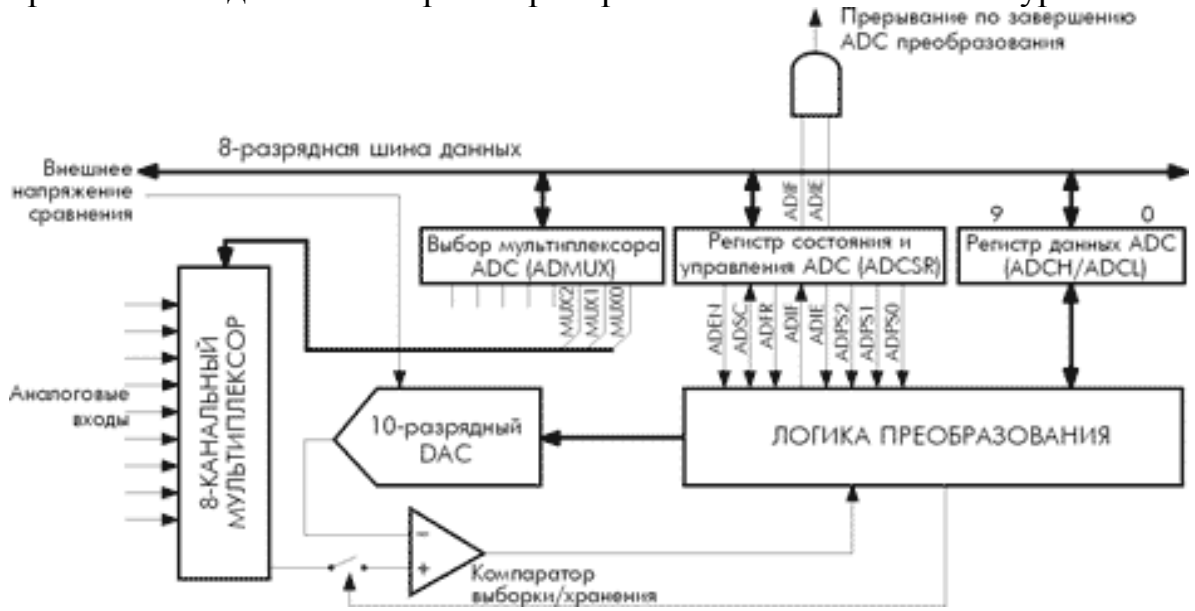


Рис. 2. Блок-схема аналого-цифрового преобразователя

Внешнее опорное напряжение подается на вывод AREF и должно быть в диапазоне от 2,7 В до AVCC.

Работа аналого-цифрового преобразователя

Аналого-цифровой преобразователь может работать в двух режимах: режиме однократного преобразования и режиме циклического преобразования. В режиме однократного преобразования каждое преобразование инициируется пользователем. В режиме циклического преобразования ADC осуществляет выборку и обновление содержимого регистра данных ADC непрерывно. Выбор режима производится битом ADFR регистра ADCSR.

Работа ADC разрешается установкой в состояние 1 бита ADEN в регистре ADCSR. Первому преобразованию, начинающемуся после разрешения ADC, предшествует пустое инициализирующее преобразование. На пользователе это отражается лишь тем, что первое преобразование будет занимать 27 тактовых циклов, вместо обычных 14.

Преобразование начинается с установки в состояние 1 бита начала преобразования ADSC. Этот бит находится в состоянии 1 в течение всего цикла преобразования и сбрасывается, по завершении преобразования, аппаратно. Если в процессе выполнения преобразования выполняется смена канала данных, то ADC вначале закончит текущее преобразование и лишь потом выполнит переход к другому каналу.

Поскольку ADC формирует 10-разрядный результат, то по завершении преобразования результирующие данные размещаются в двух регистрах данных ADCH и ADCL. При считывании данных первым должен быть считан регистр ADCL, а затем ADCH.

ADC имеет свое собственное прерывание, которое может быть активировано по завершению преобразования.

ADC работает с тактовой частотой в диапазоне от 50 до 200 кГц. В режиме циклического преобразования для преобразования необходимо 14 тактовых циклов, т.е. преобразование выполняется за время от 70 до 280 мс. В режиме однократного преобразования преобразование выполняется за 15 тактовых циклов. Если тактовая частота выйдет за указанные пределы, то правильность результата не гарантируется. Биты ADPS0 - ADPS2 используются для обеспечения необходимого диапазона тактовой частоты ADC при частоте XTAL свыше 100 кГц.

Регистр выбора мультиплексора ADC ADMUX (ADC Multiplexer Select Register)

Биты	7	6	5	4	3	2	1	0	
$07 ($27)	**REFS1**	**REFS0**	**ADLAR**	-	-	**MUX2**	**MUX1**	**MUX0**	**ADMUX**
Чтение/Запись	R	R	R	R	R	R/W	R/W	R/W	
Начальное состояние	0	0	0	0	0	0	0	0	

- **Bits 7,6 - REFS: Reference Selection Bits – Выбор опорного напряжения:**
  00 – Напряжение питания микроконтроллера
  01 – Внешнее напряжение на PB0
  10 – Внутренний источник опорного напряжения 2,56 В (отключен от PB0)
  11 – Внутренний источник опорного напряжения 2,56 В (подлючен к PB0)
- **Bits 5 - ADLAR: ADC Left Adjust Result – Выбор способа выравнивания результата**
- **Bits 4,3 - Зарезервированные биты.** Эти биты в микроконтроллерах зарезервированы и при считывании всегда покажут состояние 0.
- **Bits 2..0 - MUX2..MUX0: Analog Channel Select Bits 2-0 - Биты выбора аналогового канала.** Состояние данных битов определяет какой из аналоговых каналов будет подключен к ADC:
  001 - ADC1
  010 – ADC2
  011 – ADC3

Регистр управления и состояния ADC – ADCSR
(ADC Control and Status Register)

Биты	7	6	5	4	3	2	1	0	
$06 ($26)	**ADEN**	**ADSC**	**ADFR**	**ADIF**	**ADIE**	**ADPS2**	**ADPS1**	**ADPS0**	**ADCSR**
Чтение/Запись	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	
Нач. состоян.	0	0	0	0	0	0	0	0	

Разряд	Название	Описание
7	ADEN	Разрешение АЦП («1» — включено, «0» — выключено)
6	ADSC	Запуск преобразования («1» — начать преобразование)
5	ADFR	Выбор режима работы АЦП («0» — одиночное преобразование)
4	ADIF	Флаг прерывания от компаратора
3	ADIE	Разрешение прерывания от компаратора
2...0	ADPS2:ADPS0	Выбор частоты преобразования

- **Bit 7 - ADEN: ADC Enable - Разрешение ADC.** Установка данного бита в состояние 1 разрешает ADC. Очистка бита запрещает ADC. Запрещение ADC в процессе преобразования прекращает преобразование.
- **Bit 6 - ADSC: ADC Start Conversion - Запуск преобразования ADC.** В режиме однократного преобразования для запуска каждого цикла преобразования необходимо устанавливать бит ADSC в состояние 1. В циклическом режиме бит ADSC устанавливается в состояние 1 только при запуске первого цикла преобразования. Каждый раз после первой установки бита ADSC, выполненной после разрешения ADC или одновременно с разрешением ADC, будет выполняться пустое преобразование, предшествующее активируемому преобразованию.. Это пустое преобразование активирует ADC. ADSC будет сохранять состояние 1 в течение всего цикла преобразования и сбрасывается по завершении преобразования. При выполнении пустого преобразования, предшествующего активируемому, бит ADSC остается установленным до завершения активируемого преобразования. Запись 0 в этот бит эффекта не оказывает.

- **Bit 5 - ADFR: ADC Free Run Select - Установка циклического режима работы ADC.** При установленном в состояние 1 бите ADFR ADC будет работать в циклическом режиме. В этом режиме ADC производит выборки и обращения к регистрам непрерывно (одно за другим). Очистка бита приводит к прекращению циклического режима.
- **Bit 4 -ADIF: ADC Interrupt Flag - Флаг прерывания ADC.** Данный бит устанавливается в состояние 1 по завершению преобразования и обновления регистров данных. Прерывание по завершению преобразования ADC выполняется если в состояние 1 установлены бит ADIE и I-бит регистра SREG. Бит ADIF сбрасывается аппаратно при выполнении подпрограммы обработки соответствующего вектора прерывания. Кроме того, бит ADIF может быть очищен записью во флаг логической 1. Этого необходимо остерегаться при чтении-модификации-записи ADCSR, поскольку может быть запрещено отложенное прерывание. Это применимо и в случаях использования команд SBI и CBI.
- **Bit 3 - ADIE: ADC Interrupt Enable - Разрешение прерывания ADC.** При установленных в состояние 1 бите ADIE и I-бите регистра SREG активируется прерывание по завершению преобразования ADC.
- **Bits 2..0 - ADPS2..ADPS0: ADC Prescaler Select Bits - Выбор коэффициента предварительного деления.** Данные биты определяют коэффициент деления частоты XTAL для получения необходимой тактовой частоты ADC.

Таблица . Выбор коэффициента предварительного деления.

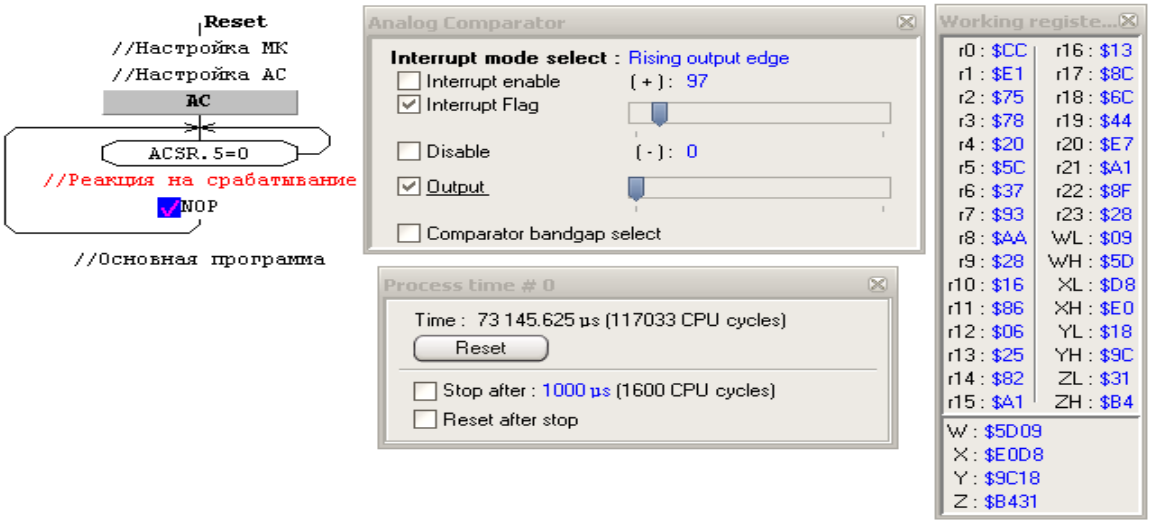
ADPS2	ADPS1	ADPS0	Коэффициент деления
0	0	0	Без деления
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Регистры данных ADC - ADCL и ADCH – (ADC Data Register)

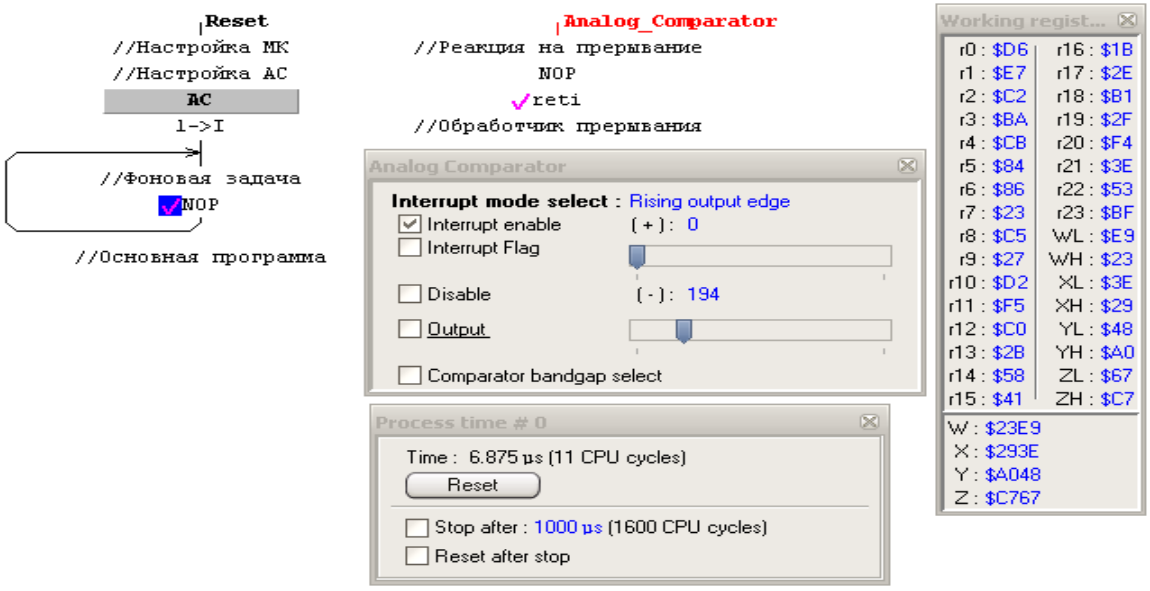
Биты	15	14	13	12	11	10	9	8	
$05 ($25)	-	-	-	-	-	-	ADC9	ADC8	ADCH
$04 ($24)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Начальное состояние	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

По завершении цикла преобразования результат преобразования размещается в этих двух регистрах. Важно, чтобы в циклическом режиме считывались оба регистра и чтобы регистр ADCL считывался перед считыванием ADCH.

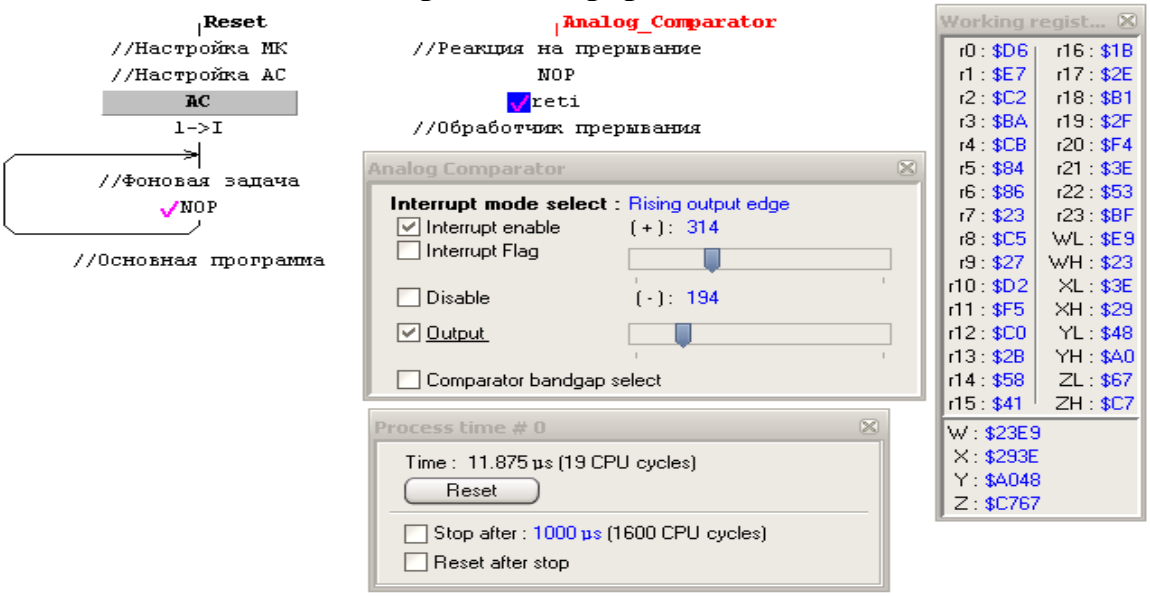
Окно отладки программы с использованием компаратора без прерываний



Окно отладки программы с использованием компаратора по прерываниям



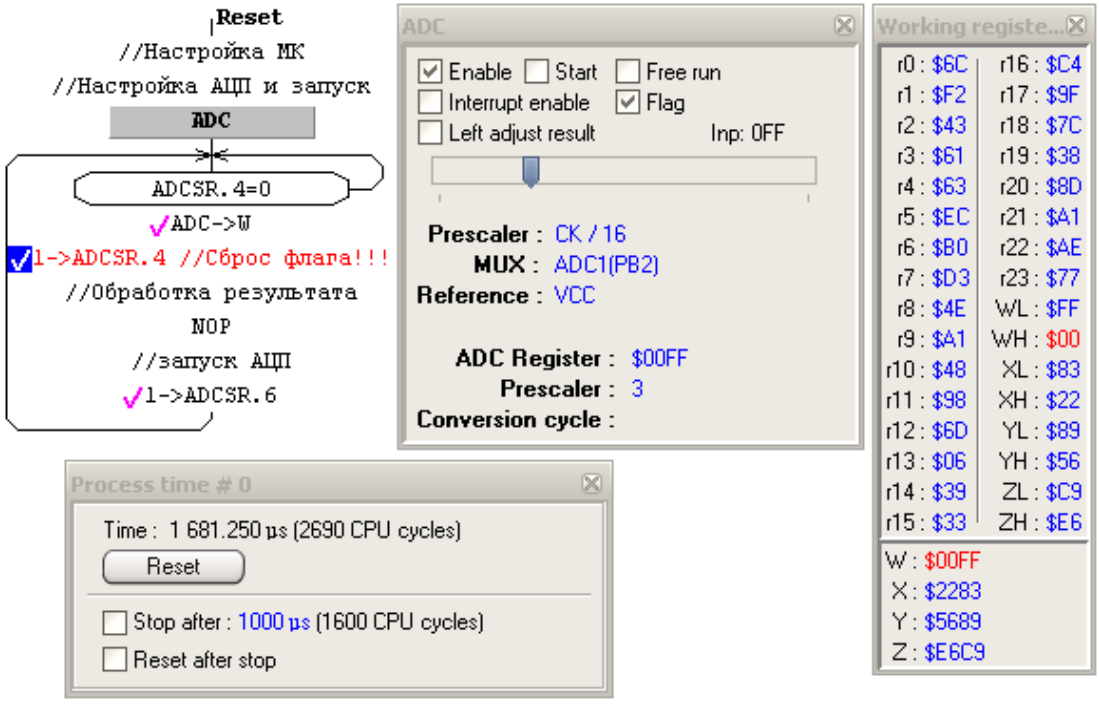
Обработка прерывания



Окно отладки программы АЦП без прерываний
с выравниванием результата вправо

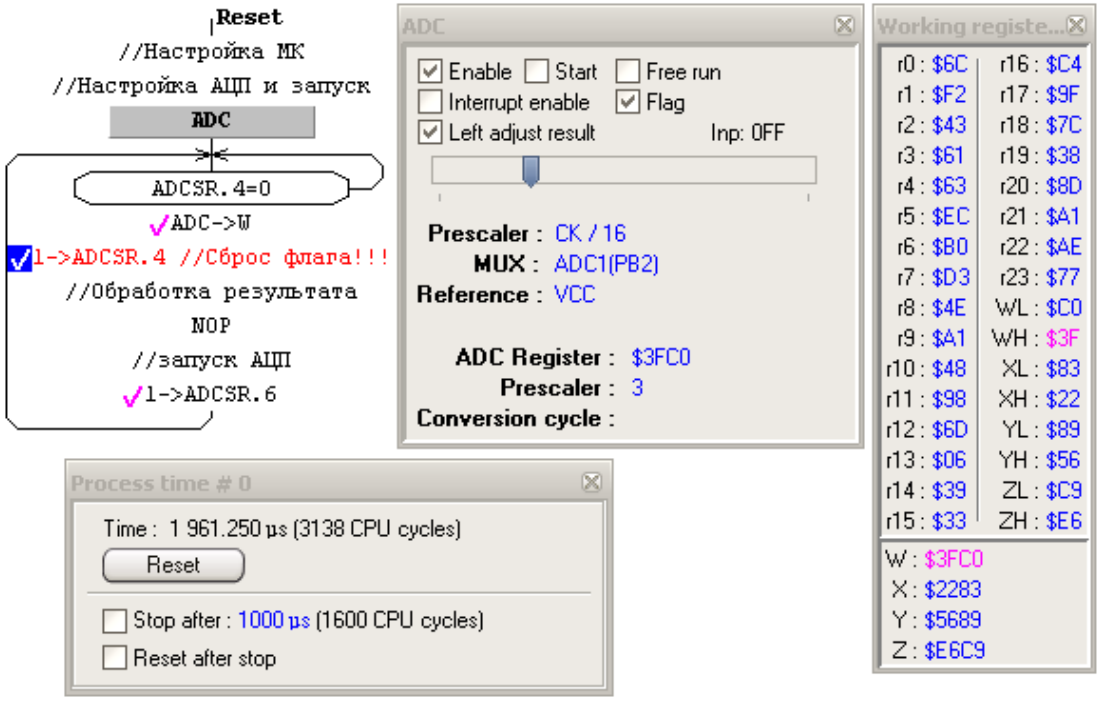
С помощью движка установлено входное напряжение U, соответствующее коду N=$0FF=0011111111$_2$,т.е.

U=Vcc/1024*N=5000мв/1024*255=1245 мв= 1,245В



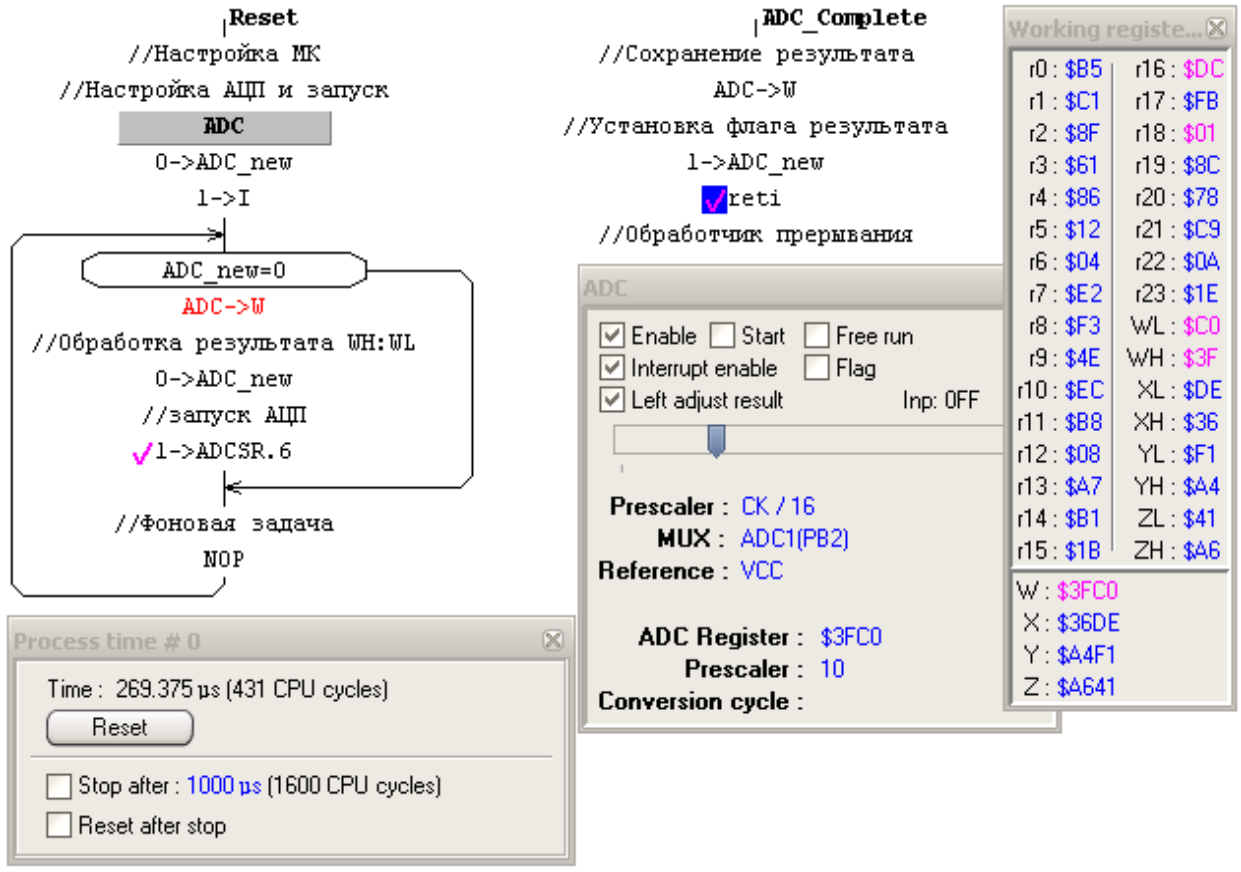
W=ADC=00000000.11111111$_2$=$00FF

Окно отладки программы АЦП без прерываний
с выравниванием результата влево



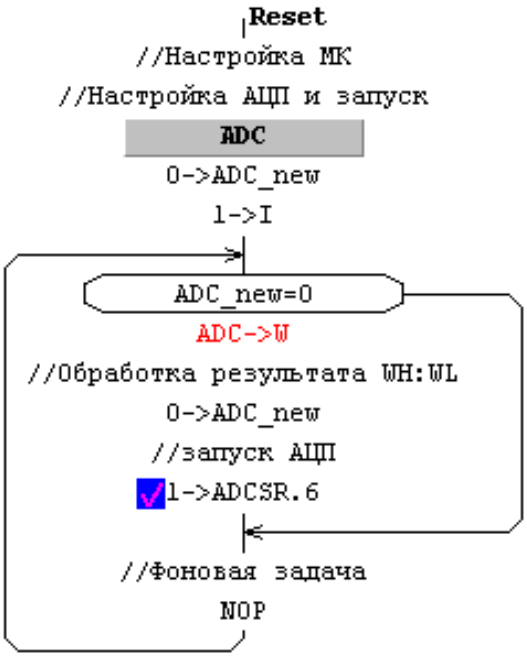
W=ADC=00111111.11000000$_2$=$3FC

Окно отладки программы АЦП с использованием прерывания и выравниванием результата влево



W=ADC=00111111.11000000$_2$=$3FC

Обработка результата после выхода из прерывания, сброс программного флага ADC_new и запуск следующего цикла прерывания.

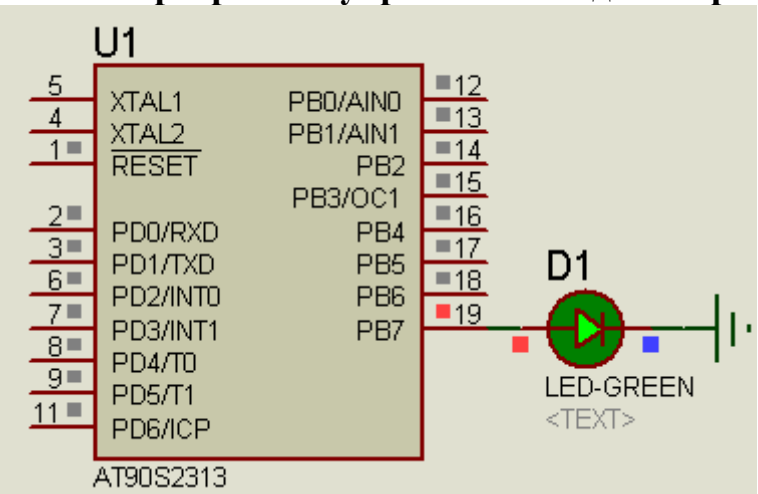


ЛАБОРАТОРНАЯ РАБОТА №9
РАЗРАБОТКА ПРОГРАММ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ С++ В СРЕДЕ WINAVR

Задание на работу:

1. Ознакомиться с примером разработки программы управления светодиодным индикатором.
2. Воспроизвести пример и провести его отладку в AVR Studio.
3. Воспроизвести пример разработки программы управления шаговым двигателем с выбором направления вращения и провести его отладку в AVR Studio.
4. Оформить отчет по работе.

1. Пример разработки программы управления индикатором

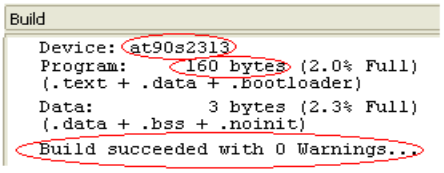


Создание проекта

1. Пуск- Все программы- Atmel AVR Tools- **AVR Studio 4**
2. New Project
3. AVR GCC
4. Location: D:\ WinAVR\ (создать папку WinAVR на диске D: , т.к. компилятор не может работать с папкой, имя которой содержит кириллицу, пробелы и т.д.)
5. Project name: Led_PB7_2313 (установить флажки Create initial file, Create folder)
6. Next
7. Debug platform: AVR Simulator
8. Device: AT90S2313
9. Finish
10. В открывшемся окне редактирования с заголовком D:\WinAVR\Led_PB7_2313\ Led_PB7_2313.c вводим текст программы:

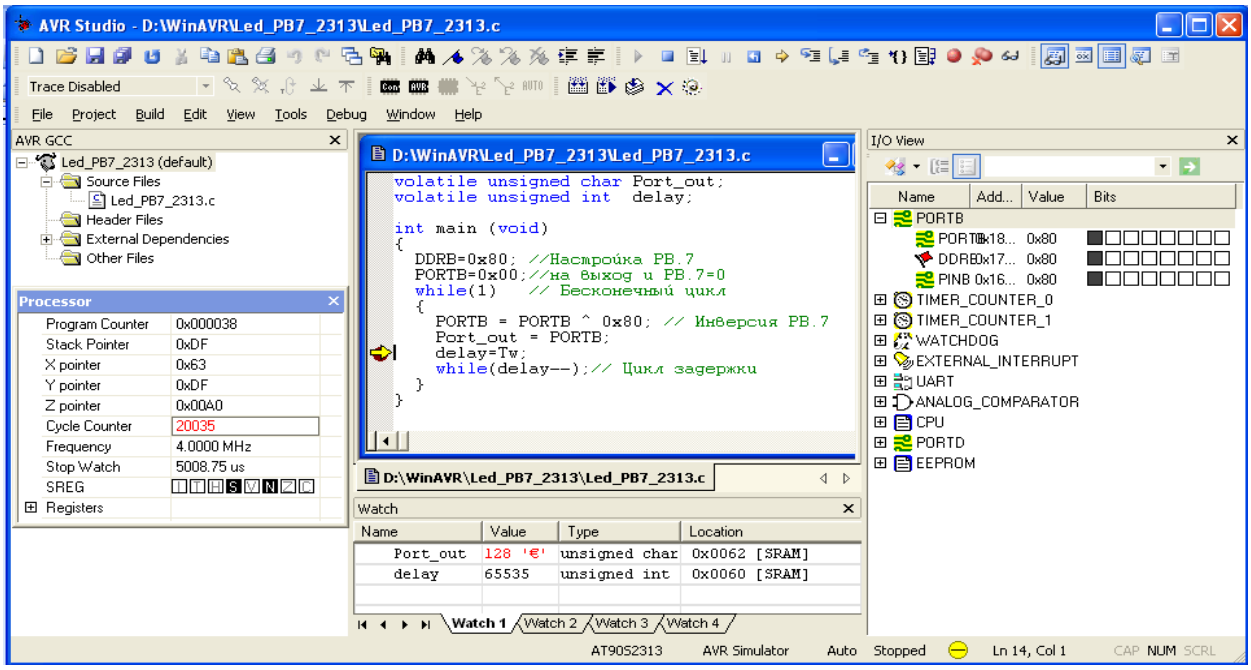
```
#include <avr/io.h>
#define  Tw  10
volatile unsigned char Port_out;
volatile unsigned int  delay;

int main (void)
{
 DDRB=0x80; //Настройка PB.7
 PORTB=0x00;//на выход и PB.7=0
 while(1)   // Бесконечный рабочий цикл
 {
    PORTB = PORTB ^ 0x80; // Инверсия PB.7
    Port_out = PORTB; // Копия PORTB
    delay=Tw;
     while(delay--);// Цикл задержки
 }
}
```

11. File- Save (сохранить файл Led_PB7_2313.c)
12.  Build Active Configuration (F7). После компиляции в окне Build выводится отчет со сведениями об ошибках и предупреждениях, либо информация о длине программы при успешном компилировании (Program: 160 bytes, или 80 слов – объем занимаемой Flash – памяти программ; Data: 3 bytes – объем SRAM- памяти (ОЗУ), занимаемой объявленными переменными Port_out (1 байт) и delay (2 байта).



13.  Build and Run (Ctrl + F7) – компиляция программы и вызов симулятора – отладчика.
14. Настройка симулятора. View – Toolbars – Processor – вызов окна процессора для контроля указателей, флагов SREG , счетчика циклов и времени выполнения программы с учетом выбранной тактовой частоты микроконтроллера. В окне I/O View открываем PORTB.
15. В нужных местах устанавливаем точки останова  (F9).
16.  (Alt+1) – открываем окно Watch для просмотра значения переменных. Активизируем меню окна Watch правой кнопкой мыши, выбираем Add Item и вносим в список имя переменной. Кнопкой Enter фиксируем переменную в списке. Аналогично в список окна Watch заносятся все необходимые переменные.

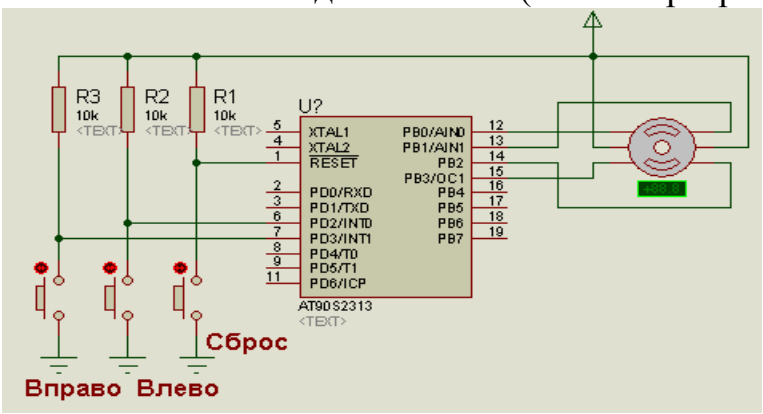
17. Выбирая «горячие» кнопки пошагового  или автоматического выполнения  до точек останова, выполнения  до курсора отлаживаемой программы, следим за изменением состояния порта, переменных и временем выполнения программы. С помощью  можно приостановить выполнение программы.
18. Для выхода из симулятора выбираем  (Ctrl + Shift + F5) или в меню Debug – Stop Debugging.



Вид рабочих окон симулятора AVR Studio в процессе отладки программы Led_PB7_2313.c :

- Ведена точка останова в строке - delay=Tw;
- В окне Watch выведены текущие значения переменных.
- В окне Processor содержится значение времени выполнения цикла (20035 тактов, или 5008,75 мкс на тактовой частоте 4 Мгц).
- В окне выведено текущее состояние регистров порта PORTB.

2. Пример управления шаговым двигателем (аналог программы из ЛР4)



```
#include <avr/io.h>
#define  Tw  10
unsigned char Port_out;
unsigned int  delay;

int main (void)
{
  DDRB=0x0F; //Настройка PB.7 на выход
  Port_out = 0b00000001;//Начальное значение
  while(1)   // Бесконечный цикл
  {
    PORTB = ~Port_out; // Вывод кода
    if(!(PIND & 0b00000100))//Проверка PIND.2=0
    { //PIND.2=0
      Port_out= Port_out<<1; // Сдвиг влево
      if(Port_out>0b00001000) Port_out= 0b00000001;
    }
    else
    if(!(PIND & 0b00001000))//Проверка PIND.3=0
    {  //PIND.3=0
      Port_out= Port_out>> 1; // Сдвиг вправо
      if(Port_out==0) Port_out= 0b00001000;
    }
    delay=Tw;
    while(delay--);// Цикл задержки
  }
}
```

```
#include <avr/io.h>
#define  Tw  10
unsigned char Port_out;
unsigned int  delay;

int main (void)
{
 DDRB=0x0F; //
 Port_out = 0b00000001;//
 while(1)   //
 {
        PORTB = ~Port_out; //
        if(!(PIND & 0b00000100))//
        { //PIND.2=0
          Port_out= Port_out<<1;
          if(Port_out>0b00001000) Port_out= 0b00000001;
  }
        else
        if(!(PIND & 0b00001000))//
        {  //PIND.3=0
          Port_out= Port_out>> 1;
          if(Port_out==0) Port_out= 0b00001000;
  }
        delay=Tw;
        while(delay--);//
 }

}
```

Содержание отчета

1. Порядок создания и отладки программы на С в WinAVR
2. Тексты программ
3. Оценка временных параметров программ
4. Сравнительная оценка длины программ в Algorithm Builder и WinAVR.

СПИСОК ЛИТЕРАТУРЫ

1. Евстифеев А.В. Микроконтроллеры AVR семейств Tiny и Mega фирмы «ATMEL». – 2-е изд., стер. –- М.: Издательский дом «Додэка-XXI», 2005. – 560 с.

2. Баранов В.Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы. – М.: Издательство дом «Додэка-XXI», 2004. – 288 с.

3. Голубцов М.С. Микроконтроллеры AVR: от простого к сложному. – М.: СОЛОН-Пресс, 2003. – 288 с.

4. Трамперт В. Измерение, управление и регулирование с помощью AVR-микроконтроллеров /Пер. с нем. – К.: «МК-Пресс», 2006. – 208 с.

5. Мортон Дж. Микроконтроллеры AVR: Вводный курс /Пер. с англ. – М.: Издательский дом «Додэка-XXI», 2006. – 272 с.

6. Костров Б. В., Ручкин В. Н. Архитектура микропроцессорных систем. – М.: Издательство Диалог-МИФИ, 2007. – 304 с.

7. Александриди Т.М., Котович И.С., Матюхина Е.Н. Организация ЭВМ и систем. Часть 4. Микропроцессорные устройства: Учебное пособие. – М.: МАДИ (ГТУ), 2008. – 68 с.

8. Бродин В.Б., Калинин А.В. Системы на микроконтроллерах и БИС программируемой логики. – М.: Издательство ЭКОМ, 2002. – 400 с.

9. Гусев В.Г. Электроника и микропроцессорная техника: Учеб. Для вузов / В.Г. Гусев, Ю.М. Гусев. – 3-е изд. – М.: Высш. шк., 2005. – 790 с.

10. Исмагилов Ф.Р., Ахматнабиев Ф.С. Микропроцессорные устройства релейной защиты энергосистем: учебное пособие / Ф.Р. Исмагилов, Ф.С. Ахматнабиев / Уфимск. гос. авиац. техн. ун-т. – Уфа: УГАТУ, 2009. – 171 с.

11. Калабеков Б.А. Цифровые устройства и многопроцессорные системы: Учебник для техникумов связи. – Горячая линия. – Телеком, 2003. – 336 с.

12. Новиков Ю.В. Основы микропроцессорной техники: учебное пособие / Ю.В. Новиков, П.К. Скоробогатов. – 4-е изд. – М.: БИНОМ. Лаборатория знаний, 2009. – 357 с.

13. Мікропроцесорна техніка: Підручник / Ю.І. Якименко, Т.О. Терещенко, Є.І. Сокол, В.Я. Жуйков, Ю.С. Петергеря; За ред. Т.О. Терещенко. – 2-ге вид. – К.: Кондор, 2004. – 440 с.

14. Яценков В.С. Микроконтроллеры Microchip. Практическое руководство. – М.: Горячая линия – Телеком, 2002. – 296 с.

15. Предко М. Справочник по PIC-микроконтроллерам. – М.: ДОДЭКА-XXI, 2002. – 512 с.

16. Таверньe К. PIC-микроконтроллеры. Практика применения. – М.: ДМК Пресс, 2002. – 272 с.

17. Файловий архів ХНАДУ: http://files.khadi.kharkov.ua/mekhatroniki-transportnikh-zasobiv/informatiki/item/8468-konspekt-lektsii.html. Нарожний В.В., Назаров О.С. Конспект лекцій з дисципліни "Мікропроцесорні пристрої". – 82 с.

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з курсу
«Мікропроцесорні пристрої»
для студентів спеціальності 6.05020103
«Комп'ютерні системи управління рухомими об'єктами»

Укладачі: О. С. НАЗАРОВ

Відповідальний за випуск О.О. Подоляка
Редактор

План 2015, поз.
Підп. до друку_____ Формат 60х84 1/16.
Умовн. друк. арк. Облік.-вид. арк. ___
Замовлення № _____Тираж ____ прим. Ціна договірна

ХНАДУ, 61002, м. Харків-МСП, вул. Петровського, 25

Свідоцтво державного комітету інформаційної політики, телебачення та радіомовлення України про внесення суб'єкта видавничої справи до державного реєстру видавців, виготівників і розповсюджувачів видавничої продукції, серія ДК, №407

Підготовлено і надруковано видавництвом Харківського національного автомобільно-дорожнього університету