*Лабораторная работа* 3

**Применение микроконтроллеров**

**для создания сложных систем**

**(Операции ввода-вывода на МК ATmega16)**

**Цель работы**

Изучить основные приёмы построения сложных систем, управляемых внешними сигналами и выдающих заданные сигналы согласно заложенному алгоритму.

Освоить методику проектирования сложной системы, создания на языке ассемблера программы управления системой и реализации системы на микроконтроллере ATmega16. Научиться программировать операции ввода-вывода на ATmega16.

Сложные системы, управляющиеся внешними сигналами и выдающие заданные сигналы согласно заложенному алгоритму, наиболее удобно реализовывать с помощью использования микроконтроллеров. Причина такого удобства состоит в универсальности физической реализации сложной системы – все системы строятся фактически по одной электронной схеме, различие между ними состоит только в разном программном обеспечении. Модификация алгоритма работы системы не требует радикального физического

|  |  |
| --- | --- |
| изменения физической структуры системы, необходима только смена программы, на основе которой система функционирует.  Алгоритм работы любой сложной системы проще всего описать графом. Система имеет несколько состояний (узлы графа), переходы от одного состояния к другому вызываются теми или иными внешними сигналами (ветви графа). Если предполагается несколько внешних сигналов (импульс таймера, сигнал датчика, нажатие кнопки и т.д.), текущее состояние системы может сменяться тем или иным состоянием, то есть ветви графа должны иметь разные цвета.  Рассмотрим в качестве примера сложную систему, граф которой приведён на рис. 1. Система имеет 10 состояний – 0, 1, …, 8, 9. При нажатии кнопки 1 (красные ветви графа) состояние системы увеличивается на 1, причём из состояния 9 система переходит в состояние 0. При нажатии кнопки 2 (зелёные ветви |  |
| Рис. 1. Пример графа сложной системы |

графа) состояние системы уменьшается на 1, причём из состояния 0 система переходит в состояние 9. При нажатии кнопки 3 (коричневые ветви графа) состояние системы увеличивается на 2, причём из состояния 8 система переходит в состояние 9, а состояние 9 не изменяется. При нажатии кнопки 4 (синие ветви графа) состояние системы уменьшается на 2, причём из состояния 1 система переходит в состояние 0, а состояние 0 не изменяется.

Проектирование любой сложной системы следует начинать с построения схемы алгоритма его работы.

|  |  |
| --- | --- |
| Схема алгоритма работы (блок-схема) рассматриваемой сис-темы приведена на рис. 2. Перио-дически производится считывание двоичного четырёхразрядного чис-ла **K**, которое задаётся четырьмя независимыми ключами. Если счи-танное значение не отличается от предыдущего (**K = Kold**), никаких действий не производится. Если оно отличается от предыдущего, но равно 0 (ни один ключ не замкнут), то также не производится никаких действий, кроме запоминания зна-чения 0. Если же при предыдущем значении **K**, равном 0, считывается новое значение, отличное от 0 (это может быть только один замкнутый ключ, а все остальные – разомкну-ты), начинается изменение состоя-ния системы. При нажатии первого ключа (**K = 0001**) текущее состояние системы увеличивается на 1, за исключением перехода 9→0 (крас-ные стрелки на рис. 1), при нажатии второго ключа (**K = 0010**) текущее |  |
| Рис. 2. Схема алгоритма работы системы |

состояние системы уменьшается на 1, за исключением перехода 0→9 (зелёные стрелки на рис. 1), при нажатии третьего ключа (**K = 0100**) текущее состояние системы увеличивается на 2, за исключением переходов 8→9 и 9→9 (коричневые стрелки на рис. 1), при нажатии четвёртогого ключа (**K = 1000**) текущее состояние системы уменьшается на 2, за исключением переходов 1→0 и 0→0 (синие стрелки на рис. 1).

Таким образом, алгоритм, реализованный по схеме рис. 2, должен обеспечивать функционирование системы, заданной графом рис. 1.

После определения схемы алгоритма следует перейти к созданию программы на языке ассемблера микроконтроллера ATmega16.

При создании сложной системы необходимо каким-либо образом организовать ввод информации в систему и вывод информации из системы. Для этой цели все микроконтроллеры снабжаются портами ввода-вывода. Микроконтроллер ATmega16 имеет 4 8-разрядных порта ввода-вывода: A, B, C и D. Каждый разряд каждого порта программируется независимо (в т.ч. некоторые разряды одного порта могут работать на ввод, а некоторые – на вывод информации), если содержимое порта используется как 8-разрядное двоичное число (например, для арифметических операций), то самым младшим является разряд Pn0 (n = A, B, C, D) а самым старшим – разряд Pn7.

Для управления разрядами каждого из регистров микроконтроллер ATmega16 имеет три регистра: PINn, PORTn и DDRn (n = A, B, C, D) разряды которых соответствуют разрядам портов.

Если DDRnk = 1, то k-ый разряд порта n работает на вывод, разряд PORTnk может принимать значения 0 либо 1, которые можно считывать или использовать для управления внешними устройствами.

Если DDRnk = 0, то k-ый разряд порта n работает на ввод, и в него можно с помощью внешних устройств записывать 0 либо 1. Для ATmega16 это означает – соединить DDRnk с 0 В либо с 5 В. Если сигнал 0 В или 5 В подаётся с какого-либо источника, ввод информации не вызывает никаких трудностей. Если же 0 В либо 5 В подаются на DDRnk посредством коммутации DDRnk с источниками постоянного напряжения, то при этом возникает обычная проблема – как ATmega16 будет воспринимать DDRnk, «висящий в воздухе» (Z-состояние). В этом случае, кроме того, возникает сильная зависимость потенциала входа DDRnk от внешних помех.

Для устранения возможности возникновения Z-состояния в микроконтроллере ATmega16 используются подтягивающие резисторы (pull-up resictors), через которые рассматриваемый вход имеет постоянное соединение с постоянным потенциалом (обычно это 0 В или 5 В). В ATmega16 при DDRnk = 0 k-ый разряд порта n в случае PORTnk = 0 имеет Z-состояние, а в случае PORTnk = 1 – соединяется через подтягивающий резистор (10 кОм) с источником питания. Поэтому вводить информацию в разряд PORTnk посредством коммутирующих устройств следует одним из способов, показанных на рис. 3.

|  |
| --- |
|  |
| Рис. 3. Варианты кнопочного ввода информации в ATmega16 |

Таким образом, прежде всего необходимо назначить один из портов микроконтроллера на ввод информации и один из портов – на вывод. Это реализуется посредством команд:

ldi temp,0x00 ; 0 --> temp

out ddrd,temp ; Назначаем порт rd на ввод (00000000 --> ddrd)

ldi temp,0xFF ; 0xff --> temp

out ddrb,temp ; Назначаем порт rb на вывод (11111111 --> ddrb)

out portd,temp ; Подключаем подтягивающие резисторы (11111111 --> portd)

Затем можно приступать непосредственно к программированию схемы алгоритма рис. 2:

|  |  |
| --- | --- |
| ldi kold, 0x00 ; 0--->kold  ldi s\_\_\_, 0x00 ; 0--->s\_\_\_  out portb, s\_\_\_ ; Вывели s\_\_\_(=0)  read: ;  in k\_\_\_, pind ; Считали содержимое порта pd (--->k\_\_\_)  cp k\_\_\_, kold ; Сравнили k\_\_\_ и kold  breq read ; Если k\_\_\_=kold, read  tst kold ; Проверили kold  brne remem ; Если kold!=0, remen  tst k\_\_\_ ; Проверили k\_\_\_  breq remem ; Если k\_\_\_=0, remem |  |

По завершению этого набора команд переменная **K** (k\_\_\_) имеет одно из значений: 1 (00012), 2 (00102), 4 (01002) или 6 (10002). Кроме этого, в переменной **Kold** (kold) хранится предыдущее значение **K**.

Различные значения переменной **K** обрабатываются в различных блоках программы:

|  |  |
| --- | --- |
| lbl1: ;  cpi k\_\_\_, 0x01 ; Сравнили k\_\_\_ и 1  brne lbl2 ; Если k\_\_\_!=1, lbl2  cpi s\_\_\_, 0x09 ; Сравнили s\_\_\_ и 9  brne lbl\_1 ; Если s\_\_\_!=9, lbl\_1  clr s\_\_\_ ; 0--->s\_\_\_  jmp print ; Перешли на Вывод  lbl\_1: ;  inc s\_\_\_ ; s\_\_\_+1--->s\_\_\_  jmp print ; Перешли на Вывод |  |

|  |  |
| --- | --- |
| lbl2: ;  cpi k\_\_\_, 0x02 ; Сравнили k\_\_\_ и 2  brne lbl4 ; Если k\_\_\_!=2, lbl4  cpi s\_\_\_, 0x00 ; Сравнили s\_\_\_ и 0  brne lbl\_2 ; Если s\_\_\_!=0, lbl\_2  ldi s\_\_\_, 0x09 ; 9--->s\_\_\_  jmp print ; Перешли на Вывод  lbl\_2: ;  dec s\_\_\_ ; s\_\_\_-1--->s\_\_\_  jmp print ; Перешли на Вывод |  |
| lbl4: ;  cpi k\_\_\_, 0x04 ; Сравнили k\_\_\_ и 4  brne lbl8 ; Если k\_\_\_!=4, lbl8  cpi s\_\_\_, 0x08 ; Сравнили s\_\_\_ и 8  brlo lbl\_4 ; Если s\_\_\_<8, lbl\_4  ldi s\_\_\_, 0x09 ; 9--->s\_\_\_  jmp print ; Перешли на Вывод  lbl\_4: ;  inc s\_\_\_ ; ...  inc s\_\_\_ ; s\_\_\_+2--->s\_\_\_  jmp print ; Перешли на Вывод |  |
| lbl8: ;  cpi k\_\_\_, 0x08 ; Сравнили k\_\_\_ и 8  brne remem ; Если k\_\_\_!=8, remem  cpi s\_\_\_, 0x02 ; Сравнили s\_\_\_ и 2  brsh lbl\_8 ; Если s\_\_\_>=2, lbl\_8  ldi s\_\_\_, 0x00 ; 0--->s\_\_\_  jmp print ; Перешли на Вывод  lbl\_8: ;  dec s\_\_\_ ; ...  dec s\_\_\_ ; s\_\_\_-2--->s\_\_\_ |  |

После завершения одного из блоков программы сформированное состояние системы **S** (s\_\_\_) выводится на порт вывода:

|  |  |
| --- | --- |
| print:  out portb,s\_\_\_ ; Вывели s\_\_\_ в порт pb  jmp remem |  |

Кроме того, текущее состояние переменной **K** запоминается в переменной **Kold**:

|  |  |
| --- | --- |
| remem: ;  mov kold, k\_\_\_ ; Записали k\_\_\_ в kold  jmp read ; Вернулись в read |  |

Таким образом, полный текст программы на языке ассемблера ATMEGA16 выглядит следующим образом:

.include "m16def.inc" ; подключение библиотеки для работы с ATmega16

.list ; включение листинга

.def temp=r16 ; определение главного рабочего регистра

.def kold=r17

.def k\_\_\_=r18

.def s\_\_\_=r19

;--------------------------------------------

.cseg ; выбор сегмента программного кода

.org 0 ; установка текущего адреса на ноль

;--------------------------------------------

ldi temp,0x80 ; выключение компаратора

out acsr,temp

;--------------------------------------------

ldi temp,0x00 ; 0 --> temp

out ddrd,temp ; Назначаем порт rd на ввод (00000000 --> ddrd)

ldi temp,0xFF ; 0xff --> temp

out ddrb,temp ; Назначаем порт rb на вывод (11111111 --> ddrb)

out portd,temp ; Подключаем подтягивающие резисторы (11111111 --> portd)

;---------------------------------------------

ldi kold, 0x00 ; 0--->kold

ldi s\_\_\_, 0x00 ; 0--->s\_\_\_

out portb, s\_\_\_ ; Вывели s\_\_\_(=0)

;---------------------------------------------

read: ;

in k\_\_\_, pind ; Считали содержимое порта pd (--->k\_\_\_)

cp k\_\_\_, kold ; Сравнили k\_\_\_ и kold

breq read ; Если k\_\_\_=kold, read

tst kold ; Проверили kold

brne remem ; Если kold!=0, remen

tst k\_\_\_ ; Проверили k\_\_\_

breq remem ; Если k\_\_\_=0, remem

jmp lbl1 ;

;---------------------------------------------

remem: ;

mov kold, k\_\_\_ ; Записали k\_\_\_ в kold

jmp read ; Вернулись в read

;---------------------------------------------

lbl1: ;

cpi k\_\_\_, 0x01 ; Сравнили k\_\_\_ и 1

brne lbl2 ; Если k\_\_\_!=1, lbl2

cpi s\_\_\_, 0x09 ; Сравнили s\_\_\_ и 9

brne lbl\_1 ; Если s\_\_\_!=9, lbl\_1

clr s\_\_\_ ; 0--->s\_\_\_

jmp print ; Перешли на Вывод

lbl\_1: ;

inc s\_\_\_ ; s\_\_\_+1--->s\_\_\_

jmp print ; Перешли на Вывод

;---------------------------------------------

lbl2: ;

cpi k\_\_\_, 0x02 ; Сравнили k\_\_\_ и 2

brne lbl4 ; Если k\_\_\_!=2, lbl4

cpi s\_\_\_, 0x00 ; Сравнили s\_\_\_ и 0

brne lbl\_2 ; Если s\_\_\_!=0, lbl\_2

ldi s\_\_\_, 0x09 ; 9--->s\_\_\_

jmp print ; Перешли на Вывод

lbl\_2: ;

dec s\_\_\_ ; s\_\_\_-1--->s\_\_\_

jmp print ; Перешли на Вывод

;---------------------------------------------

lbl4: ;

cpi k\_\_\_, 0x04 ; Сравнили k\_\_\_ и 4

brne lbl8 ; Если k\_\_\_!=4, lbl8

cpi s\_\_\_, 0x08 ; Сравнили s\_\_\_ и 8

brlo lbl\_4 ; Если s\_\_\_<8, lbl\_4

ldi s\_\_\_, 0x09 ; 9--->s\_\_\_

jmp print ; Перешли на Вывод

lbl\_4: ;

inc s\_\_\_ ; ...

inc s\_\_\_ ; s\_\_\_+2--->s\_\_\_

jmp print ; Перешли на Вывод

;---------------------------------------------

lbl8: ;

cpi k\_\_\_, 0x08 ; Сравнили k\_\_\_ и 8

brne remem ; Если k\_\_\_!=8, remem

cpi s\_\_\_, 0x02 ; Сравнили s\_\_\_ и 2

brsh lbl\_8 ; Если s\_\_\_>=2, lbl\_8

ldi s\_\_\_, 0x00 ; 0--->s\_\_\_

jmp print ; Перешли на Вывод

lbl\_8: ;

dec s\_\_\_ ; ...

dec s\_\_\_ ; s\_\_\_-2--->s\_\_\_

;---------------------------------------------

print: ;

out portb, s\_\_\_ ; Вывели s\_\_\_ в порт pb

jmp remem

Написанную программу следует оттранслировать с помощью программы AVR\_Studio. Созданный программой hex-файл необходимо связать с микроконтроллером ATmega16 в системе PROTEUS. При сборке разработанной системы в программе PROTEUS удобно использовать для отображения состояния системы семисегментный индикатор типа 7SEG-BCD, а в качестве ключей для подачи информации на вход микроконтроллера – кнопки BUTTON, как это показано на рис. 4.

|  |
| --- |
|  |
| Рис. 4. Реализация разработанной системы в программе PROTEUS |

Следует отметить, что кнопка BUTTON имеет по умолчанию следующие параметры: сопротивление в нажатом состоянии – 100 мОм, сопротивление в ненажатом состоянии – 100 МОм, то есть при нажатии на неё, действительно наступает замыкание, как это и требуется в правой схеме рис. 3. В левой же схеме того же рисунка необходимо, чтобы нажатие на кнопку вызывало размыкание цепи. Это можно легко обеспечить, переназначив значения этих сопротивлений – например, 100 МОм в нажатом состоянии и 100 мОм – в ненажатом.

**Порядок выполнения работы**

1. Получить у преподавателя задание – граф переходов системы

2. Составить схему алгоритма работы системы.

3. Написать на языке ассемблера ATmega16 программу, реализующую алгоритм п.2.

4. С помощью программы AVR\_Studio осуществить трансляцию программы (получить hex-файл).

5. Собрать в системе PROTEUS схему на основе микроконтроллера ATmega16, реализующую разрабатываемую систему.

6. Убедиться в правильном функционировании разработанной системы.

**Содержание отчёта**

Отчёт должен содержать:

1. Задание лабораторной работы – граф переходов разрабатываемой системы.

2. Алгоритм работы системы

3. Программу для микроконтроллера ATmega16, реализующую разработанный алгоритм.

4. Функционирование разработанной системы должно быть продемонстрировано в программе PROTEUS.