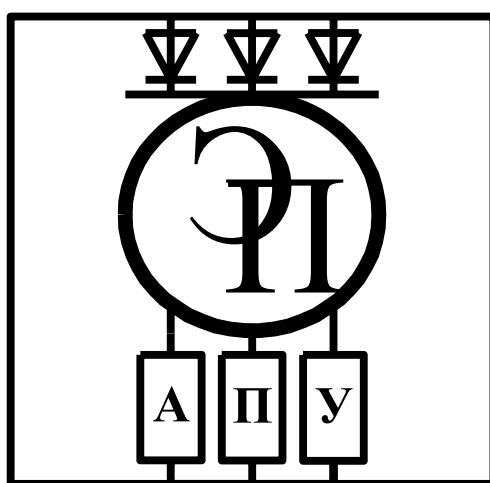


МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Электропривод и автоматизация промышленных установок»

МИКРОПРОЦЕССОРНАЯ ТЕХНИКА В МЕХАТРОНИКЕ И РОБОТОТЕХНИКЕ

*Методические рекомендации к практическим занятиям
для студентов специальности
15.03.06 «Мехатроника и робототехника»
очной формы обучения*



Могилев 2023

УДК 004.4:621.865.8
ББК 32.973.2:32.816
М59

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Электропривод и автоматизация промышленных установок» «27» сентября 2022 г., протокол № 2

Составители: ст. преподаватель О. А. Капитонов;
Д. В. Шнип

Рецензент Д. М. Свирепа

Методические рекомендации предназначены для студентов направления подготовки 15.03.06 «Мехатроника и робототехника» дневной формы обучения.

Учебно-методическое издание

МИКРОПРОЦЕССОРНАЯ ТЕХНИКА
В МЕХАТРОНИКЕ И РОБОТОТЕХНИКЕ

Ответственный за выпуск	С. М. Фурманов
Корректор	Т. А. Рыжикова
Компьютерная верстка	М. М. Дударева

Подписано в печать 08.02.2023 . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. 2,56. Уч.-изд. л. 2,63 .Тираж 36 экз. Заказ № 155 .

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».

Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.

Пр-т Мира, 43, 212022, г. Могилев.

© Белорусско-Российский
университет, 2023

Содержание

Введение.....	4
1 Практическое занятие № 1. Арифметические основы вычислительной техники.....	5
2 Практическое занятие № 2. Представление и обработка информации в микропроцессорных системах	10
3 Практическое занятие № 3. Логические основы вычислительной техники	14
4 Практическое занятие № 4. Применение цифровых элементов в микропроцессорных системах	17
5 Практическое занятие № 5. Применение аналоговых элементов в микропроцессорных системах	23
6 Практическое занятие № 6. Составление алгоритмов и программ, реализующих ввод и обработку дискретных сигналов	27
7 Практическое занятие № 7. Составление алгоритмов и программ, реализующих ввод и обработку аналоговых сигналов.....	30
8 Практическое занятие № 8. Составление алгоритмов и программ, реализующих временные функции управления.....	34
9 Практическое занятие № 9. Составление алгоритмов и программ, реализующих типовые функции обработки информации.....	40
Список литературы	42

Введение

Целью преподавания дисциплины «Микропроцессорная техника в мехатронике» является получение студентами знаний о принципах построения микропроцессорных систем.

Студент, изучающий дисциплину, знает арифметические и логические основы микропроцессорной техники, основные логические элементы и узлы, используемые в микропроцессорных устройствах; принципы программного управления обработкой информации и особенности представления информации в микропроцессорных системах; назначение, принцип действия, структуру и функциональные особенности микропроцессоров и других элементов микропроцессорных систем; классификацию микропроцессоров и их параметры; структуру микропроцессорных систем и особенности реализации отдельных элементов; перспективы развития микропроцессорной техники и её влияние на развитие современного автоматизированного производства.

Студент, изучивший дисциплину, умеет производить выбор структуры микропроцессорной системы и средств для её реализации, осуществлять разработку типовых алгоритмов и программ обработки информации в микропроцессорных устройствах; владеет методикой исполнения современных средств для разработки и отладки программного обеспечения микропроцессорных систем.

Методические рекомендации составлены на основе [1].

1 Практическое занятие № 1. Арифметические основы вычислительной техники

Цель занятия: овладеть методикой перевода чисел и выполнения арифметических операций над ними в разных системах счисления.

Задание

Для задач, выданных преподавателем, выполнить перевод чисел, а также произвести над ними ряд арифметических операций.

1.1 Системы счисления

Микропроцессор может обрабатывать только информацию, представленную в числовой форме. Система счисления – это способ записи чисел с помощью заданного набора специальных знаков (цифр).

Запись числа в некоторой системе счисления называется кодом числа. В общем случае число записывается следующим образом:

$$X = a_n a_{n-1} \dots a_0. \quad (1.1)$$

Отдельную позицию в записи числа принято называть разрядом, а номер позиции – номером разряда, количество разрядов в записи числа – это разрядность, и она совпадает с длиной числа.

Если при записи числа значимость («вес») цифры зависит от местоположения (позиции), занимаемого в числе, то такая система счисления называется позиционной. Произвольное число X в позиционной системе с основанием q в общем виде можно записать в виде полинома

$$X_q = \sum_{i=-m}^n a_i q^i = a_n q^n + a_{n-1} q^{n-1} + \dots + a_0 q^0 + a_{-1} q^{-1} + \dots + a_{-m} q^{-m}, \quad (1.2)$$

где a_i – разрядный коэффициент $a_i = 0 \dots q-1$;
 q^i – весовой коэффициент.

Число q называется основанием системы счисления, оно равно количеству различных знаков или символов, используемых для изображения цифр в данной системе. Разряды с положительными степенями q образуют целую часть числа X_q , а с отрицательными – дробную. Цифры a_n и a_{-m} соответственно являются старшим и младшим разрядами числа.

В зависимости от целей применения используют различные системы счисления: двоичную, восьмеричную, десятичную, шестнадцатеричную. В двоичной системе всего две цифры – 0 и 1, в десятичной – 10 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), в восьмеричной системе – 8 (0, 1, 2, 3, 4, 5, 6, 7), в шестнадцатеричной – 16 (в качестве первых 10 из 16 шестнадцатеричных цифр взяты привычные цифры 0, 1, ..., 9, а вот в качестве остальных шести цифр используют первые буквы

$$\begin{array}{r|l}
 & x \\
 0,28125 & 8 \\
 2\ 25 & 8 \\
 2\ 0 &
 \end{array}$$

Получим $0,28125_{10} = 0,22_8$.

Правило 3. При переводе числа из любой системы счисления в десятичную удобно пользоваться разложением числа в виде многочлена по степеням основания.

Например, переведем число 111011_2 в десятичную систему счисления:

$$111011_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 59_{10}.$$

Правило 4. При переводе числа из восьмеричной или шестнадцатеричной системы счисления в двоичную каждую восьмеричную или шестнадцатеричную цифру заменяют соответствующей триадой (для восьмеричной системы) или тетрадой (для шестнадцатеричной системы).

Например, переведем числа $537,1_8$ и $1A3,F_{16}$ в двоичную систему счисления:

$$537,1_8 = 101\ 011\ 111,001_2,$$

$$1A3,F_{16} = 1\ 1010\ 0011,1111_2.$$

Правило 5. При переводе из двоичной системы счисления в восьмеричную или шестнадцатеричную исходное число разбивают (вправо и влево от запятой) на триады (для восьмеричной) или тетрады (для шестнадцатеричной) и каждую полученную триаду или тетраду заменяют соответствующей ей цифрой в восьмеричной или шестнадцатеричной системе.

Например, переведем число $10010011111,1110011_2$ в шестнадцатеричную систему счисления:

$$10010011111,1110011_2 = 0100\ 1001\ 1111,1110\ 0110 = 49F, E6_{16}.$$

В таблице 1.1 приведены соответствие кодов некоторых чисел в различных системах счисления.

Таблица 1.1

Двоичный	Восьмеричный	Десятичный	Шестнадцатеричный
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4

Окончание таблицы 1.1

Двоичный	Восьмеричный	Десятичный	Шестнадцатеричный
101	5	5	5
110	6	6	6
111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F

1.2 Арифметические операции в позиционных системах счисления

Рассмотрим основные арифметические операции: сложение, вычитание, умножение и деление. Правила выполнения этих операций в десятичной системе хорошо известны это сложение, вычитание, умножение столбиком и деление углом. Эти правила применимы и ко всем другим позиционным системам счисления.

Сложение чисел выполняется поразрядно, начиная с младшего разряда. Получившееся в результате число не должно превышать основания системы счисления, поэтому если при сложении возникает избыток, то он переносится влево и учитывается при сложении чисел в следующем разряде. При этом результат сложения уменьшается на основание системы счисления.

Например, при сложении двух чисел X и Y в двоичной системе счисления коды суммы C и переноса Π формируются в соответствии с таблицей 1.2.

Таблица 1.2

X	Y	Π	C
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Пример 1 – Сложим числа 0111_2 и 0101_2 :

$$\begin{array}{r} 0111 \\ + \\ 0101 \\ \hline 1100 \end{array}$$

Ответ: 1100_2 .

Аналогичные правила действуют и для других чисел, представленных в других системах счисления.

Пример 2 – Сложим числа 15 и 6 в шестнадцатеричной системе счисления:

$$15_{10} + 6_{10} = F_{16} + 6_{16} = 15_{16} = 25_8 = 10101_2 = 21_{10}.$$

Ответ: 15_{16} .

Для проверки преобразуем полученные суммы к десятичному виду:

$$10101_2 = 2^4 + 2^2 + 2^0 = 16 + 4 + 1 = 21,$$

$$25_8 = 2 \cdot 8^1 + 5 \cdot 8^0 = 16 + 5 = 21,$$

$$15_{16} = 1 \cdot 16^1 + 5 \cdot 16^0 = 16 + 5 = 21.$$

Вычитание также производится поразрядно, начиная с младшего разряда. При вычитании большего числа из меньшего необходимо произвести заем из старшего разряда, что равносильно добавлению числа, равного основанию системы счисления, к уменьшаемому числу. Заем при этом уменьшает следующий разряд на 1.

Пример 3 – Вычтем единицу из чисел 100_2 , 100_8 и 10_{16} :

$$\begin{array}{r} \dot{1}00 \\ -001 \\ \hline 011 \end{array} \quad \begin{array}{r} \dot{1}00 \\ -001 \\ \hline 077 \end{array} \quad \begin{array}{r} \dot{1}0 \\ -01 \\ \hline 0F \end{array}$$

Ответ: $100_2 - 1_2 = 11_2$, $100_8 - 1_8 = 77_8$, $10_{16} - 1_{16} = F_{16}$.

Выполняя умножение многозначных чисел в различных позиционных системах счисления, можно использовать обычный алгоритм перемножения чисел в столбик, но при этом результаты перемножения и сложения однозначных чисел необходимо заимствовать из соответствующих рассматриваемой системе таблиц умножения и сложения.

Ввиду чрезвычайной простоты таблицы умножения в двоичной системе умножение сводится лишь к сдвигам множимого и сложениям.

Пример 4 – Перемножим числа 101101_2 и 101_2 :

$$\begin{array}{r} 101101 \\ \times \quad 101 \\ \hline 101101 \\ 000000 \\ 101101 \\ \hline 11100001 \end{array}$$

Ответ: $101101_2 \cdot 101_2 = 45_{10} \cdot 5_{10} = 225_{10} = 11100001_2$.

Деление в любой позиционной системе счисления производится по тем же правилам, как и деление углом в десятичной системе. В двоичной системе деление выполняется особенно просто, ведь очередная цифра частного может быть только нулем или единицей.

Пример 5 – Разделим число 101101_2 на число 101_2 .

$$\begin{array}{r} \underline{101101} \quad | \quad 101 \\ \underline{101} \quad \quad 1001 \\ \quad \quad \underline{0101} \\ \quad \quad \quad \underline{101} \\ \quad \quad \quad \quad \underline{0} \end{array}$$

2 Практическое занятие № 2. Представление и обработка информации в микропроцессорных системах

Цель занятия: изучить способы представления некоторых типов информации в микропроцессорных устройствах.

Задание

Записать карту памяти микропроцессорной системы для данных, полученных от преподавателя.

2.1 Общие сведения о представлении информации

Микропроцессор в основных современных приложениях может обрабатывать числовую, текстовую, графическую, звуковую и видеоинформацию. Все эти виды информации для процессора представляются в двоичном коде, т. е. используется алфавит мощностью два (всего два символа 0 и 1). Связано это с тем, что удобно представлять информацию в виде последовательности электрических импульсов: сигнал отсутствует (0), сигнал есть (1). Такое кодирование принято называть двоичным, а сами логические последовательности нулей и единиц – машинным языком.

Таким образом, каждая цифра машинного двоичного кода несет количество информации, равное одному биту. Будучи объединенными, биты формируют двоичное сообщение.

2.2 Представление числовой информации

Числа – основной вид информации, с которой работает микропроцессор. Значения физических величин, измеряемых системами управления, представляют собой числа.

Для записи целых чисел используется либо двоичная система счисления, либо двоично-десятичная форма записи. Как было указано ранее, в двоичной системе счисления числа записываются при помощи последовательности нулей

$$\begin{array}{r}
 00000101_2 \\
 +11111010_2 \\
 \hline
 11111011_2
 \end{array}$$

Как видно, при таком способе представления старший разряд отрицательного числа равен 1 и указывает на знак числа так же, как и в предыдущем способе. Однако отличием является то, что старший разряд в дополнительном коде имеет вес, хоть он и является отрицательным:

$$11111011 = (-1) \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = -5.$$

Дробные числа могут быть представлены двумя способами.

1 С фиксированной запятой, когда для хранения целой части числа используется n -разрядное двоичное слово, для хранения дробной части используется m -разрядное двоичное слово (рисунок 2.3).

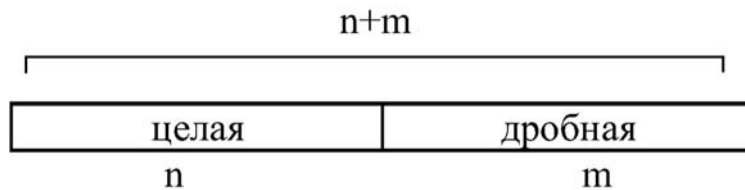


Рисунок 2.3 – Дробное число с фиксированной запятой

2 С плавающей запятой, когда для хранения мантиисы числа используется n -разрядное двоичное слово (с учетом знакового разряда), для порядка используется m -разрядное двоичное слово (с учетом знакового разряда) (рисунок 2.4).

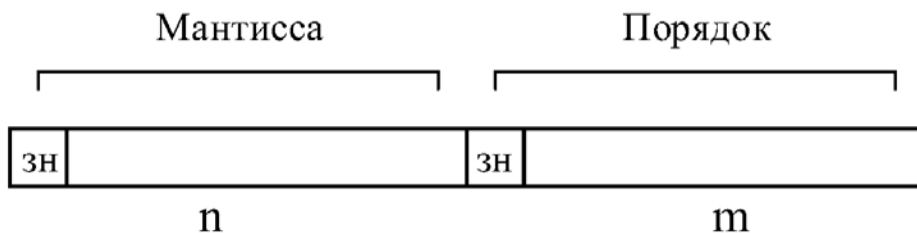


Рисунок 2.4 – Дробное число с плавающей запятой

Рассмотрим пример представления дробного числа с фиксированной запятой, при записи которого используются восемь разрядов для целой и восемь разрядов для дробной части. Для записи такого числа процессору понадобится 16 разрядов, фиксированная запятая расположена между 7-м и 8-м разрядами, (рисунок 2.5).

Например, число $45,5_{10}$ после перевода в двоичную систему счисления

принимает вид 101101_2 . Полученное число можно дополнить нулями, не имеющими веса – в старших разрядах для целой части и в младших для дробной. В результате число примет вид $00101101,10000000_2$. Целая часть этого числа будет записана в старших восьми разрядах, дробная часть – в младших.

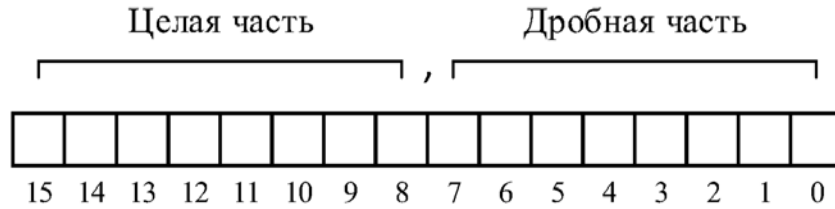


Рисунок 2.5 – 16-разрядное дробное число с фиксированной запятой

При сохранении в памяти никаких символов «,» не сохраняется, и число $45,5_{10}$ будет записано как двухбайтное число 0010110110000000_2 (рисунок 2.6).

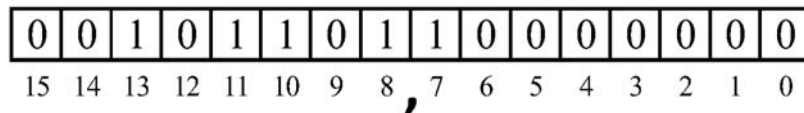


Рисунок 2.6 – 16-разрядное число $45,5$ с фиксированной запятой

2.3 Порядок следования байтов

Память микропроцессорных систем, как правило, имеет побайтовую организацию, т. е. каждая ячейка позволяет сохранять один байт или 8-битное двоичное слово. Однако, как было ранее указано, такое слово не может превышать значения 255. Числа больше 255 требуют большего количества ячеек для хранения, так, например, число 60000_{10} представляется 16-разрядным двоичным словом $1110\ 1010\ 0110\ 0000_2$, для хранения которого требуется 2 байта.

Таким образом, информация может быть представлена в виде последовательности байтов. В том случае, если число не может быть представлено одним байтом, имеет значение, в каком порядке байты записываются в памяти компьютера или передаются по линиям связи. В современной вычислительной технике наиболее распространены два способа порядка следования байтов:

- 1) little-endian – от младшего к старшему;
- 2) big-endian – от старшего к младшему.

При использовании порядка «от младшего к старшему» запись начинается с младшего и заканчивается старшим. Для порядка «от старшего к младшему», наоборот, запись начинается со старшего и заканчивается младшим. На рисунке 2.7 приведен пример размещения байтов в памяти. Из него видно, что для little-endian-порядка младшие байты многобайтного числа будут располагаться по младшим адресам, старшие байты будут располагаться в ячейках с большим адресом. Для big-endian наоборот.

Например, если необходимо расположить в памяти по адресу 0 4-байтное

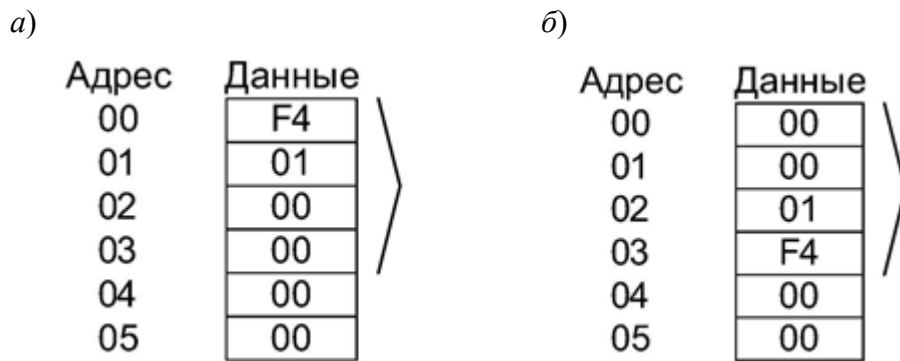
число 500_{10} , потребуется четыре ячейки с адресами 0, 1, 2 и 3. Само число в виде 4-байтного (т. е. 32-разрядного) двоичного числа будет представлено как

$$500_{10} = 000000000000000000000001\ 1111\ 0100_2 = 000001F4_{16}.$$

Такая последовательность байтов будет размещена в памяти, как указано на рисунке 2.8.



Рисунок 2.7 – Размещение байтов слова в памяти



а – порядок little-endian; б – порядок big-endian

Рисунок 2.8 – Число 500 в памяти

3 Практическое занятие № 3. Логические основы вычислительной техники

Цель занятия: изучить способы представления некоторых типов информации в микропроцессорных.

Задание

Для задач, выданных преподавателем, выполнить ряд логических операций.

Основу микропроцессорных устройств составляют элементарные логические схемы, действующие по законам и правилам алгебры логики.

Алгебра логики – раздел математики, изучающий высказывания, рассматриваемые со стороны их логических значений (истинности или ложности)

и логических операций над ними.

Высказывание – это некоторое законченное предложение, о котором можно однозначно сказать, что оно истинно или ложно.

Каждому высказыванию можно приписать одно из двух значений: истинному высказыванию соответствует значение 1, а ложному – 0.

Из нескольких простых высказываний можно составлять сложные высказывания. Для объединения простых высказываний в сложные применяются логические операции: конъюнкция, дизъюнкция, исключаящее ИЛИ, отрицание.

Конъюнкция двух высказываний A и B (логическое умножение, операция И) обозначается $A \cdot B$ или $A \wedge B$ (читается A и B). Правило выполнения: результат операции логического умножения является истинным только тогда, когда истинны оба высказывания. Если оба высказывания или хотя бы одно из них ложно, то и $A \wedge B$ тоже ложно. Таблица истинности логической операции конъюнкция приведена в таблице 3.1.

Дизъюнкция двух высказываний A и B (логическое сложение, операция ИЛИ) обозначается $A \vee B$ или $A + B$ (читается A или B). Правило выполнения: результат операции логического сложения является истинным тогда, когда истинно хотя бы одно из исходных высказываний. Таблица истинности логической операции дизъюнкция приведена в таблице 3.2.

Исключаящее ИЛИ двух высказываний A и B (сложение по модулю два) обозначается $A \oplus B$. Правило выполнения: результат операции является истинным, если число складываемых единичных битов нечетно, если же их число четно, то результат ложен. Таблица истинности логической операции исключаящее ИЛИ приведена в таблице 3.3.

Отрицание высказывания A (операция НЕ, инверсия) обозначается \bar{A} (читается «не A »). Правило выполнения: инверсия одного значения переменной совпадает с другим ее значением. Таблица истинности логической операции инверсия приведена в таблице 3.4.

Таблица 3.1

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Таблица 3.2

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Таблица 3.3

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Таблица 3.4

A	\bar{A}
0	1
1	0

Кроме того, микропроцессоры могут производить данные логические операции над группой битов, объединенных в сообщение или слово. При этом операция поочередно выполняется над равнозначными разрядами двоичного слова. Примеры выполнения логических операций над двоичными числами приведены в таблице 3.5.

Для того чтобы выполнить любую из приведенных логических операций над числами в других системах счисления (восьмеричной, десятичной,

шестнадцатеричной), их необходимо перевести в двоичный вид, выполнить операцию и перевести в исходную систему счисления.

Таблица 3.5

Операция ИЛИ	Операция И	Операция исключающее ИЛИ	Операция инверсия
$\begin{array}{r} 101101 \\ \vee \\ \underline{001110} \\ 101111 \end{array}$	$\begin{array}{r} 101101 \\ \wedge \\ \underline{001110} \\ 001100 \end{array}$	$\begin{array}{r} 101101 \\ \oplus \\ \underline{001110} \\ 100011 \end{array}$	$\begin{array}{r} \overline{1011001} = \\ = 0100110 \end{array}$

Например, для чисел 67_{10} и 114_{10} будут получены следующие результаты логических операций:

$$\begin{aligned} 67_{10} &= 0100\ 0011_2 \\ 114_{10} &= 0111\ 0010_2 \end{aligned}$$

$$\begin{array}{r} 01000011 \\ \vee 01110011 \\ \hline 01110011 \end{array}$$

$$01110011_2 = 2^6 + 2^5 + 2^4 + 2^1 + 2^0 = 115_{10}.$$

В этом случае работать с числами, записанными в восьмеричной или шестнадцатеричной системах счисления значительно проще, т. к. для перевода их в двоичный вид необходимо каждую цифру представить в виде триады (для восьмеричной) или тетрады (для шестнадцатеричной), после чего выполнить логическую операцию.

Например, для чисел $A9_{16}$ и 35_{16} логическая операция И будет выполняться следующим образом:

$$\begin{array}{r} A9 = 01000011 \\ 35 = \wedge 01110011 \\ \hline 01000011 \end{array} = 23_{16}.$$

Глядя на результат побитовых операций, не сразу можно уловить закономерности в их результате. Поэтому непонятно, зачем нужны такие операции. Однако они находят свое применение. В байтах не всегда хранятся числа. Байт или ячейка памяти может хранить набор флагов (установлен/сброшен), представляющих собой информацию о состоянии чего-либо. С помощью битовых логических операций можно проверить, какие биты в байте установлены в единицу, можно обнулить биты или, наоборот, установить в единицу. Также существует возможность сменить значения битов на противоположные.

4 Практическое занятие № 4. Применение цифровых элементов в микропроцессорных системах

Цель занятия: изучить основные типы цифровых микросхем в микропроцессорных устройствах.

Задание

Составить схемы обработки цифровой информации по заданиям, выданным преподавателем.

4.1 Цифровые элементы, реализующие логические операции в микропроцессорных устройствах

Для обработки цифровых сигналов существует множество схем, которые позволяют хранить цифровую информацию и осуществлять ее преобразование. Например, над электронными цифрами можно производить следующие операции: сложение, вычитание, умножение, деление, инвертирование, поразрядный сдвиг и др. Кроме того, существуют более сложные элементы цифровой техники: триггеры, дешифраторы, мультиплексоры, счетчики, регистры и т. д.

В настоящее время цифровые микросхемы получили такое большое распространение, что логические элементы можно встретить в схемах, очень далеких от микропроцессоров и вычислительной техники.

Из всего разнообразия цифровых элементов большинство – это составные элементы. Составные в том смысле, что их можно составить из других, более простых. А в основе лежат всего три простейших логических элемента.

Все элементы работают с цифровыми сигналами. То есть сигналы на входе должны принимать значения либо логического нуля, либо логической единицы. На выходе каждый элемент также обеспечивает цифровой сигнал, который в зависимости от логики работы схемы принимает значение либо логической единицы, либо логического нуля.

Элемент «И» и элемент «ИЛИ» на рисунке 4.1 изображены с двумя входами. На самом деле они могут иметь любое количество входов.

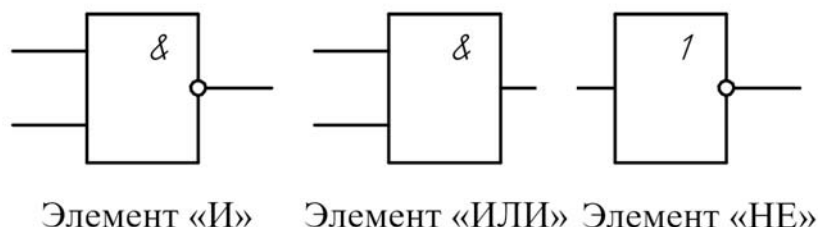


Рисунок 4.1 – Базовые цифровые элементы

Элемент «И». На выходе этого элемента сигнал логической единицы появляется только тогда, когда на всех входах будет присутствовать логическая единица. И на первом, и на втором, и на третьем. На всех имеющихся входах.

Если хотя бы на одном входе будет ноль, то и на выходе тоже будет ноль.

Элемент «ИЛИ». На выходе этого элемента появится логическая единица тогда, когда хотя бы на одном из входов появится единица. Или на первом, или на втором, или на третьем. На любом из имеющихся входов. Логический ноль на выходе будет только тогда, когда на всех входах будет сигнал логического нуля.

Элемент «НЕ». Или инвертор. У этого элемента всегда один вход и один выход. И работает он просто. Когда на входе у инвертора сигнал логического нуля, на выходе логическая единица. И наоборот, когда на входе логическая единица, на выходе логический ноль.

Собирая из этих трех основных элементов различные схемы, можно получить все разнообразие цифровых и логических элементов, применяемых в микропроцессорной технике. Такие составные элементы для удобства чтения сложных схем, в свою очередь, имеют свои схемные обозначения и выглядят на схеме как самостоятельные элементы.

На рисунке 4.2 приведены таблицы истинности для трех основных логических элементов (элементы «И» и «ИЛИ» приведены в трехвходовых вариантах).

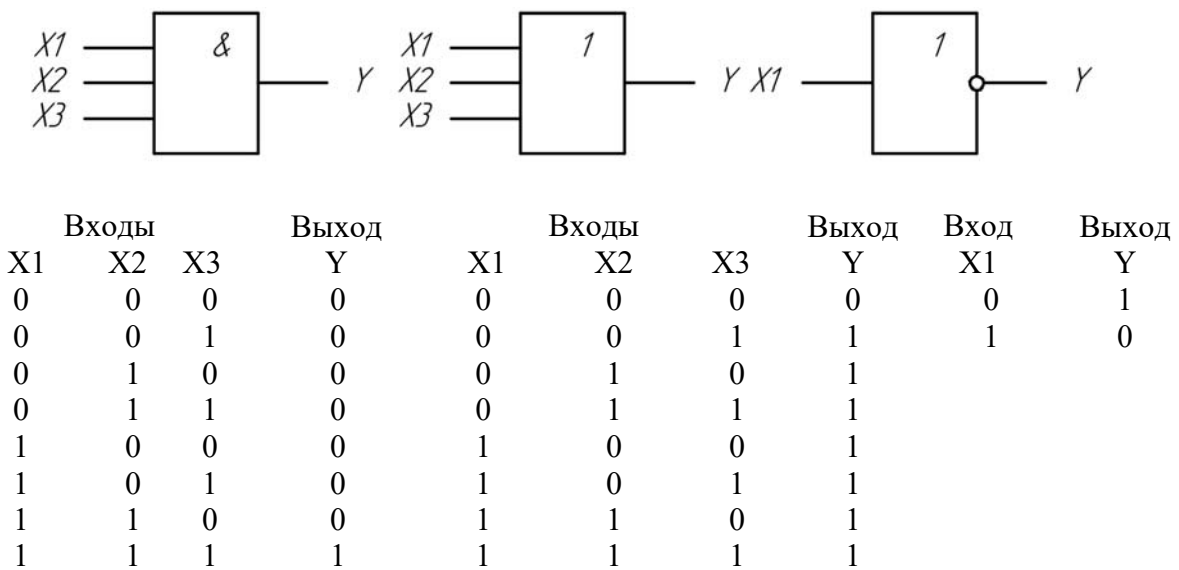
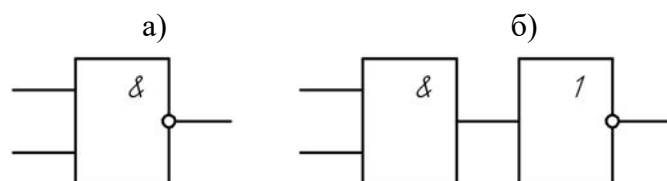


Рисунок 4.2 – Базовые логические элементы

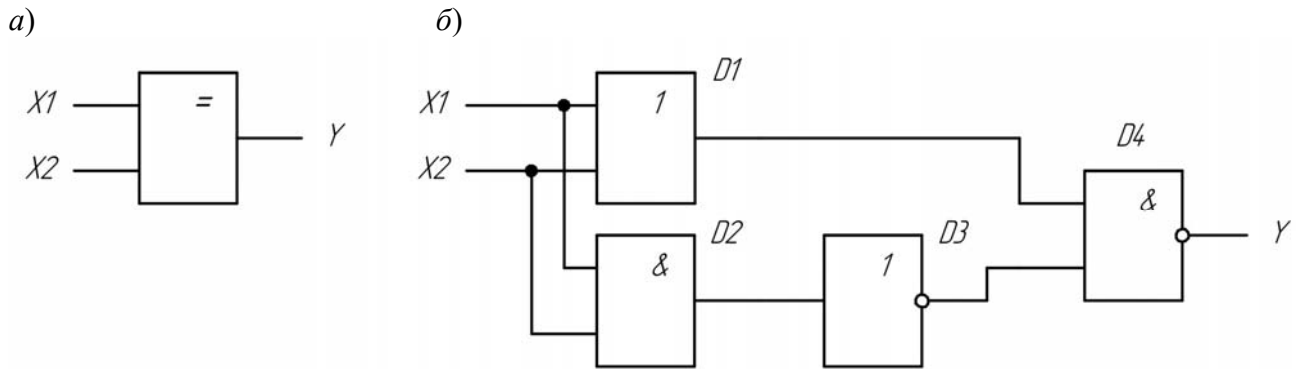
Если соединить элемент «И» с инвертором так, как это показано на схеме (рисунок 4.3), получится новый элемент – «И-НЕ».



а – условно-графическое обозначение; б – составные элементы

Рисунок 4.3 – Элемент «И-НЕ»

Элемент «ИСКЛЮЧАЮЩЕЕ ИЛИ» (рисунок 4.4) является более сложным. Его тоже можно встретить в различных электронных схемах.



а – условно-графическое обозначение; б – составные элементы

Рисунок 4.4 – Элемент «ИСКЛЮЧАЮЩЕЕ ИЛИ»

4.2 Триггеры

Триггеры – это устройства с двумя состояниями. Они предназначены для запоминания двоичной информации. Использование триггеров позволяет реализовывать устройства оперативной памяти (то есть памяти, информация в которой хранится только на время вычислений). Однако это не единственная их область применения. Триггеры широко используются для построения цифровых устройств с памятью, таких как счётчики, преобразователи последовательного кода в параллельный, последовательные порты или цифровые линии задержки, применяемые в составе цифровых фильтров.

Простейшая схема триггера, позволяющая запоминать двоичную информацию, может быть построена на двух логических инверторах, охваченных положительной обратной связью. Она приведена на рисунке 4.5.

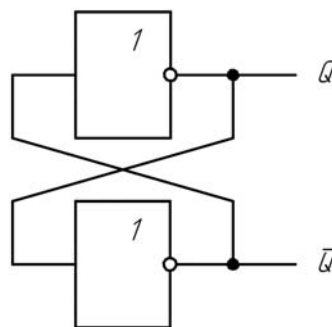
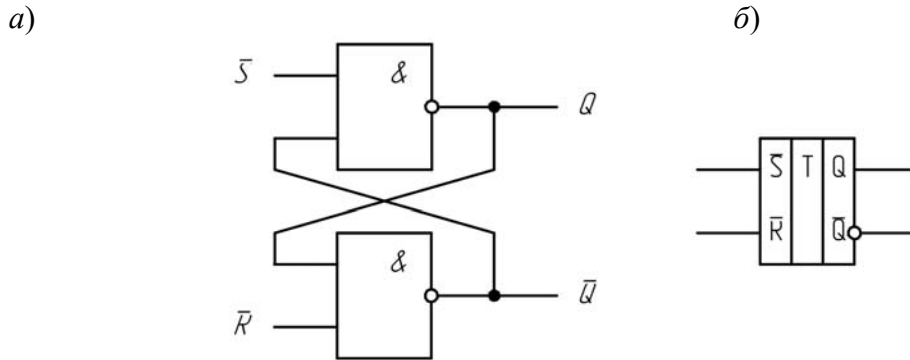


Рисунок 4.5 – Схема простейшего триггера, построенного на инверторах

В схеме любого триггера может быть только два состояния – на выходе Q присутствует логическая единица и на выходе Q присутствует логический ноль. На выводе \bar{Q} будет сигнал, противоположный Q.

RS-триггер получил название по названию своих входов. Вход S (англ. Set – установить) позволяет устанавливать выход Q в единичное состояние

(устанавливать означает записывать логическую единицу); вход R (англ. Reset – сбросить) – сбрасывать выход Q (англ. Quit – выход) в нулевое состояние. Для его реализации можно воспользоваться логическими элементами «2И-НЕ» (рисунок 4.6).



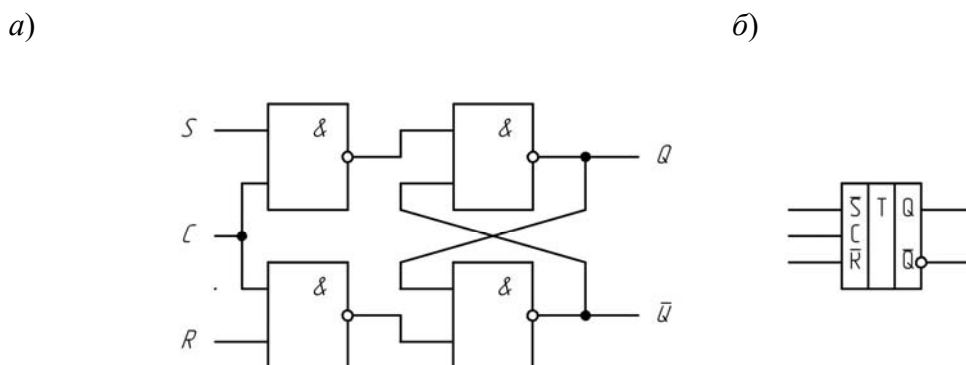
а – реализация; б – условно-графическое обозначение

Рисунок 4.6 – Схема RS-триггера на логических элементах «2И-НЕ»

Схема RS-триггера позволяет запоминать состояние логической схемы, но так как в начальный момент времени может возникать переходный, то запоминать состояния логической схемы в триггерах нужно только в определённые моменты времени, когда все переходные процессы закончены.

Это означает, что большинство цифровых схем требуют сигнала синхронизации (тактового сигнала). Все переходные процессы в комбинационной логической схеме должны закончиться за время периода синхросигнала, подаваемого на входы триггера. Триггеры, запоминающие входные сигналы только в момент времени, определяемый сигналом синхронизации, называются синхронными.

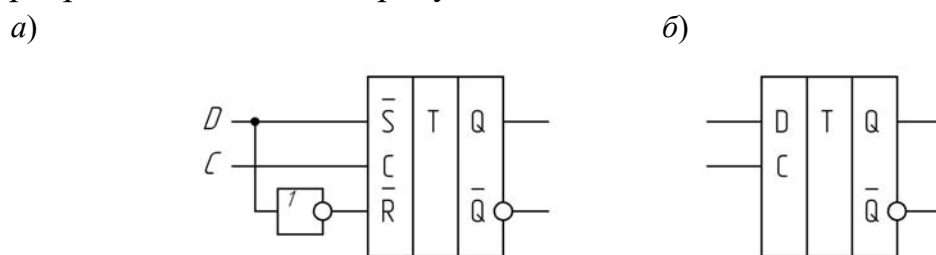
Для этого нам потребуется схема, пропускающая входные сигналы только при наличии синхронизирующего сигнала. Принципиальная схема синхронного RS-триггера приведена на рисунке 4.7.



а – реализация; б – условно-графическое обозначение

Рисунок 4.7 – Схема синхронного RS-триггера, построенного на элементах «И»

В RS-триггерах для записи логического нуля и логической единицы требуются разные входы, что не всегда удобно. При записи и хранении данных один бит может принимать значение как нуля, так и единицы. Для его передачи достаточно одного провода. Сигналы установки и сброса триггера не могут появляться одновременно, поэтому можно объединить эти входы при помощи инвертора, как показано на рисунке 4.8.



a – реализация; *б* – условно-графическое обозначение

Рисунок 4.8 – Принципиальная схема D-триггера (защелки)

Такой триггер получил название D-триггер. Конкретное значение задержки определяется частотой следования импульсов синхронизации.

4.3 Шифраторы и дешифраторы

Шифраторы (кодеры) – устройства, предназначенные для преобразования алфавитно-цифровой информации, поданной унитарным n -разрядным кодом в эквивалентный двоичный m -разрядный код. Особенностью унитарного кода является активное состояние только одной переменной X_i входного набора $\{X_{n-1} \dots X_1 X_0\}$, порядковый номер i которой подлежит кодированию. То есть шифратор n - m – это преобразователь унитарного кода «1 из n » в двоичный (параллельный) код, у которых число выходов m однозначно связано с числом входов n как 2^m . При $n = 2^m$ используется полный набор выходных двоичных комбинаций Y_i . Такой шифратор называется полным. Например, шифратор 8-3 полный, т. к. он реализует полный набор возможных комбинаций переменных X_i ($n = 8$) в полный выходной набор Y_i ($m = 3$) как $2^3 = 8$.

В неполном шифраторе число входов n не соответствует числу всех возможных выходных комбинаций 2^m ($n < 2^m$), что, соответственно, создает некоторое число неиспользованных выходных наборов. Примером неполного шифратора является шифратор 10-4, используемый для кодирования десятичных чисел в двоичный код (8-4-2-1). На рисунке 4.9 приведено условно-графическое обозначение полного восьмиразрядного шифратора, а в таблице 4.1 – его таблица истинности.

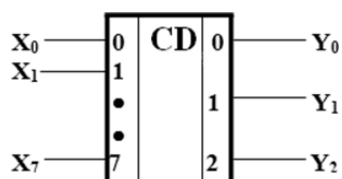


Рисунок 4.9 – Шифратор

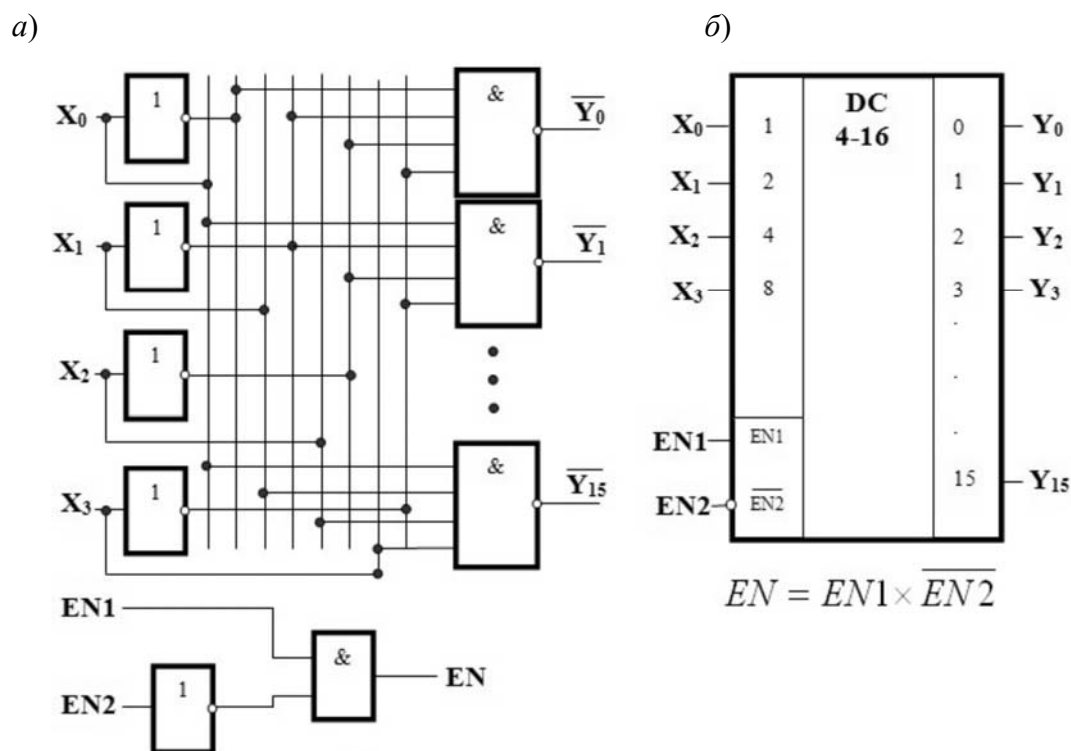
Таблица 4.1

Число	Вход								Выход		
	X ₀	X ₁	X ₂	X ₃	X ₄	X ₅	X ₆	X ₇	Y ₂	Y ₁	Y ₀
0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	0	1	0
3	0	0	0	1	0	0	0	0	0	1	1
4	0	0	0	0	1	0	0	0	1	0	0
5	0	0	0	0	0	1	0	0	1	0	1
6	0	0	0	0	0	0	1	0	1	1	0
7	0	0	0	0	0	0	0	1	1	1	1

Шифраторы применяются в устройствах, преобразующих один вид кода в другой. Шифратор может быть организован не только для представления (кодирования) десятичного числа двоичным кодом, но и для выдачи определенного кода (его значение заранее выбирается), например, при нажатии клавиши с соответствующим символом. При появлении этого кода система оповещается о том, что нажата определенная клавиша клавиатуры.

Дешифраторы (декодеры) – устройства для распознавания числа, поданного позиционным n -разрядным кодом. Дешифраторы относятся к преобразователям кодов. Дешифратор n - m выполняет функцию преобразования двоичного кода в унитарный код «1 из m », т. е. выполняет функцию, обратную шифратору.

На рисунке 4.10, *а* показана схема функциональная дешифратора, на рисунке 4.10, *б* – его условно-графическое обозначение.



а – схема функциональная; *б* – условное графическое обозначение

Рисунок 4.10. – Полный дешифратор 4-16

Для полного дешифратора $m = 2^n$, где m – порядковый номер выхода Y_i дешифратора. В неполном дешифраторе число выходов m не соответствует 2^n ($m < 2^n$).

В условном графическом обозначении (см. рисунок 4.10, б) входы дешифратора обозначают их двоичными весами. Помимо информационных входов, дешифратор обычно имеет один или более входов разрешения работы (EN). При наличии разрешения по этому входу дешифратор работает описанным выше образом, при его отсутствии – все выходы дешифратора пассивны. Если входов разрешения несколько, то сигнал разрешения работы образуется как конъюнкция сигналов отдельных входов разрешения.

Более подробные сведения о цифровых элементах микропроцессорных систем можно найти в методических указаниях к практическому занятию № 4, электронная версия которых имеется в классе ПЭВМ кафедры (лаб. 207/2) по следующему адресу: D:\методические рекомендации\СТАНДАРТЫ РФ\Специальность 15.03.06\Микропроцессорная техника в мехатронике и робототехнике\МПТ ПЗ 04.pdf.

5 Практическое занятие № 5. Применение аналоговых элементов в микропроцессорных системах

Цель занятия: изучить основные типы аналоговых микросхем в микропроцессорных устройствах.

Задание

Составить схемы обработки аналоговых сигналов по заданиям, выданным преподавателем.

Основными элементами микропроцессорной системы для работы с аналоговыми сигналами являются преобразователи «аналоговый сигнал – цифровой код». Без аналого-цифрового преобразователя (АЦП) и цифро-аналогового преобразователя (ЦАП) система не сможет обеспечить ввод-вывод аналоговых сигналов, т. к. микропроцессор является цифровым устройством.

Аналого-цифровой преобразователь (АЦП) – один из самых важных электронных компонентов в измерительном и тестовом оборудовании. АЦП преобразует напряжение (аналоговый сигнал) в код, над которым микропроцессор и программное обеспечение выполняют определенные действия. Даже если Вы работаете только с цифровыми сигналами, то, скорее всего, Вы используете АЦП в составе осциллографа, чтобы узнать их аналоговые характеристики.

Одним из основных параметров АЦП является его разрядность, которая характеризует количество дискретных значений, которые преобразователь может выдать на выходе. В двоичных АЦП разрядность измеряется в битах.

Разрядностью АЦП определяется и его разрешающая способность – величина, обратная максимальному числу кодовых комбинаций на выходе АЦП.

$$R = \frac{1}{2^N}. \quad (5.1)$$

Этот параметр определяет, какой минимальный уровень входного сигнала (относительно сигнала полной амплитуды) способен воспринимать АЦП.

Еще один параметр, характеризующий АЦП, – точность, которая определяется абсолютной погрешностью полной шкалы $\delta_{\text{ни}}$, нелинейностью и дифференциальной нелинейностью, по отношению к входному сигналу.

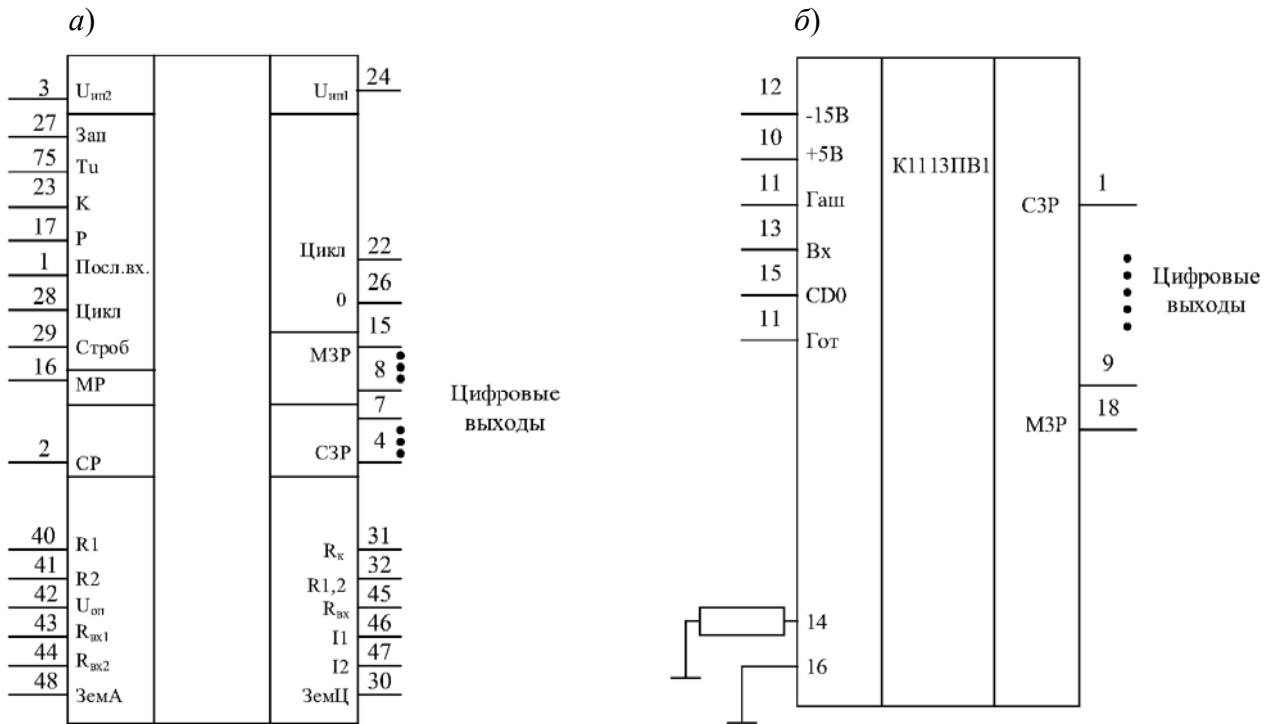
Быстродействие АЦП характеризуется временем преобразования $t_{\text{прб}}$, т. е. интервалом времени от момента заданного измерения сигнала на входе до появления на выходе установившегося кода.

Существует несколько основных типов архитектуры АЦП, отличающихся по сложности реализации, быстродействию, помехоустойчивости, хотя в пределах каждого типа существует также множество вариаций. Различные типы измерительного оборудования используют различные типы АЦП. Например, в цифровом осциллографе используется высокая частота дискретизации, но не требуется высокое разрешение. В цифровых мультиметрах нужно большее разрешение, но можно пожертвовать скоростью измерения.

Системы сбора данных общего назначения по скорости дискретизации и разрешающей способности обычно занимают место между осциллографами и цифровыми мультиметрами. В оборудовании такого типа используются АЦП последовательного приближения либо сигма-дельта АЦП. В системах, где основным критерием является быстродействие, применяют АЦП параллельного преобразования. Однако преобразователи этого типа сложны в реализации. Для n -разрядного АЦП необходимо $2^n - 1$ компараторов и параллельный делитель напряжения, который вырабатывает $2^n - 1$ уровней квантования. Кроме того, существуют также интегрирующие АЦП с высокими разрешением и помехоподавлением. Такой АЦП состоит из двух преобразователей. Измеряемое напряжение преобразуется в длительность импульса, а потом длительность импульса преобразуется в цифровой код.

В связи с многообразием внутренних структур АЦП микросхемы преобразователей отличаются друг от друга. Так, на рисунке 5.1 приведена цоколевка микросхем АЦП.

У микросхемы К572ПВ1 выводы имеют следующее назначение: 1 – цифровой последовательный вход; 2 – вход управления выходами старших разрядов; 3 – $U_{\text{ип1}}$; 4 (СЗР) ... 15 (МЗР) – цифровые входы (выходы); 16 – вход управления входами-выходами младших разрядов; 17 – вход управления режимом ЦАП-АЦП; 22 – выход «Цикл»; 23 – вход сравнения; 24 – $U_{\text{ип2}}$; 75 – вход тактовых импульсов; 26 – выход «Конец преобразования»; 27 – вход «Запуск»; 28 – вход «Цикл»; 29 – стробирование ЦАП; 36 – цифровая земля; 31 – конечный вывод матрицы R–2R; 32 – общий вывод резисторов R1 и R2; 40 и 41 – выводы резисторов R1 и R2; 42 – опорное напряжение; 43 и 44 – аналоговые входы 1 и 2; 45 – общий вывод резисторов аналоговых входов; 46 и 47 – аналоговые выходы 1 и 2; 48 – аналоговая «земля».



а – К572ПВ1; б – К1113ПВ1

Рисунок 5.1 – Микросхемы АЦП

У микросхемы К1113ПВ1 выводы имеют следующее назначение: 1 (9-й разряд) ... 9 (СЗР) – цифровые выходы; 10 – $U_{ип1}$; 11 – вход управления выводами вводом данных; 12 – $U_{ип2}$; 13 – аналоговый вход; 14 – аналоговая «земля»; 15 – управление сдвигом нуля; 16 – цифровая «земля»; 17 – выход готовности данных; 18 – МЗР; в – схема включения в МП-систему.

ЦАП – это устройство для преобразования цифрового кода в аналоговый сигнал по величине, пропорциональной значению кода.

ЦАП применяются для связи цифровых управляющих систем с устройствами, которые управляются уровнем аналогового сигнала. Также ЦАП является составной частью во многих структурах аналого-цифровых устройств и преобразователей.

ЦАП характеризуется функцией преобразования. Она связывает изменение цифрового кода с изменением напряжения или тока. Функция преобразования ЦАП выражается следующим образом:

$$U_{вых} = \frac{U_{max}}{N_{max}} N_{вх}, \quad (5.2)$$

где $U_{вых}$ – значение выходного напряжения, соответствующее цифровому коду $N_{вх}$, подаваемому на входы ЦАП;

U_{max} – максимальное выходное напряжение, соответствующее подаче на входы максимального кода N_{max} .

Принцип работы большинства ЦАП – это суммирование долей аналоговых сигналов (веса разряда) в зависимости от входного кода.

ЦАП можно реализовать с помощью суммирования токов, суммирования напряжений и деления напряжений. В первом и втором случаях в соответствии со значениями разрядов входного кода, суммируются сигналы генераторов токов и источников ЭДС. Последний способ представляет собой управляемый кодом делитель напряжения. Два последних способа не нашли широкого распространения в связи с практическими трудностями их реализации.

Кроме того, ЦАП можно классифицировать и по другим признакам:

- по роду выходного сигнала (с токовым выходом, выходом по напряжению, с изменяющимся сопротивлением);

- по полярности выходного напряжения (постоянному, переменному) и т. д.

Основные структуры, используемые в ЦАП интегрального исполнения:

- со взвешенными резисторами в цепях эмиттеров;

- со взвешенными резисторами в цепях нагрузки;

- с лестничной матрицей $R-2R$ в цепях эмиттеров транзисторов источников тока;

- с выходной лестничной матрицей $R-2R$.

Структура простейшего ЦАП с взвешенным суммированием токов приведена на рисунке 5.2.

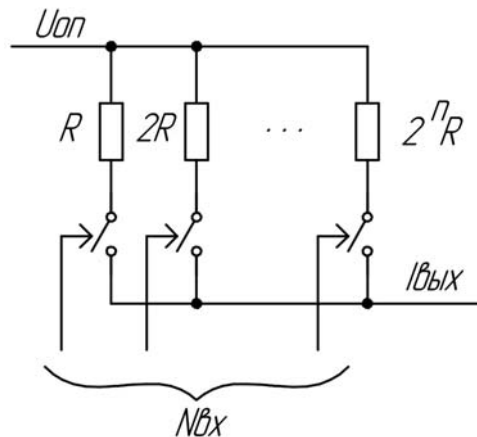


Рисунок 5.2 – Упрощенная схема ЦАП

Более подробные сведения о аналоговых элементах микропроцессорных систем можно найти в методических указаниях к практическому занятию № 5, электронная версия которых имеется в классе ПЭВМ кафедры (лаб. 207/2) по следующему адресу: D:\методические рекомендации\СТАНДАРТЫ РФ\Специальность 15.03.06\Микропроцессорная техника в мехатронике и робототехнике\МПТ ПЗ 05.pdf.

6 Практическое занятие № 6. Составление алгоритмов и программ, реализующих ввод и обработку дискретных сигналов

Цель занятия: изучить методику разработки программ для обработки дискретных сигналов.

Задание

По заданной преподавателем схеме составить блок-схему алгоритма, а также программу, производящую обработку информации.

В системах автоматизации очень распространены двоичные сигналы, которые поступают от концевых выключателей, датчиков охранной или пожарной сигнализации, датчиков заполнения емкостей, датчиков сбегания ленты на конвейере, датчиков приближения и т. п. Такие сигналы не совсем правильно называются дискретными, но этот термин прочно вошел в практику.

При вводе дискретных сигналов может понадобиться решить следующие задачи:

- запомнить значение дискретного сигнала;
- произвести гальваническую развязку;
- согласовать сигнал по уровню;
- обеспечить индикацию состояния выхода.

При выводе дискретных сигналов необходимо:

- запомнить значение дискретного сигнала;
- произвести гальваническую развязку;
- усилить сигнал по уровню;
- индикация состояния выхода.

При разработке программы для большинства микроконтроллеров, позволяющей организовать ввод-вывод дискретных сигналов, следует произвести настройку соответствующих выводов портов, к которым подключены входные и выходные цепи. Настройка портов заключается в настройке как минимум направления передачи информации через соответствующие линии портов путем записи управляющего слова в специальный регистр микроконтроллера. Формат управляющего слова, а также другие возможности зависят от типа микроконтроллера и возможностей, заложенных в порт производителем микросхемы.

Пример схемы, обеспечивающей ввод/вывод дискретных сигналов в микропроцессорной системе, который может быть использован, например, для организации пульта оператора, приведен на рисунке 6.1.

В схеме два дискретных датчика ДД1 и ДД2 оформлены в виде двух переключателей SW1 и SW2, подключенных к выводам P3.2 и P3.3 микроконтроллера. Два дискретных индикатора оформлены в виде двух светодиодов ДСИ1 и ДСИ2, подключенных через транзисторные ключи к выводам P3.5 и P3.4 микроконтроллера.

Пусть необходимо для приведенной выше системы разработать программу, позволяющую отображать на ДСИ1 состояние ДД1 и на ДСИ2 состояние ДД2.

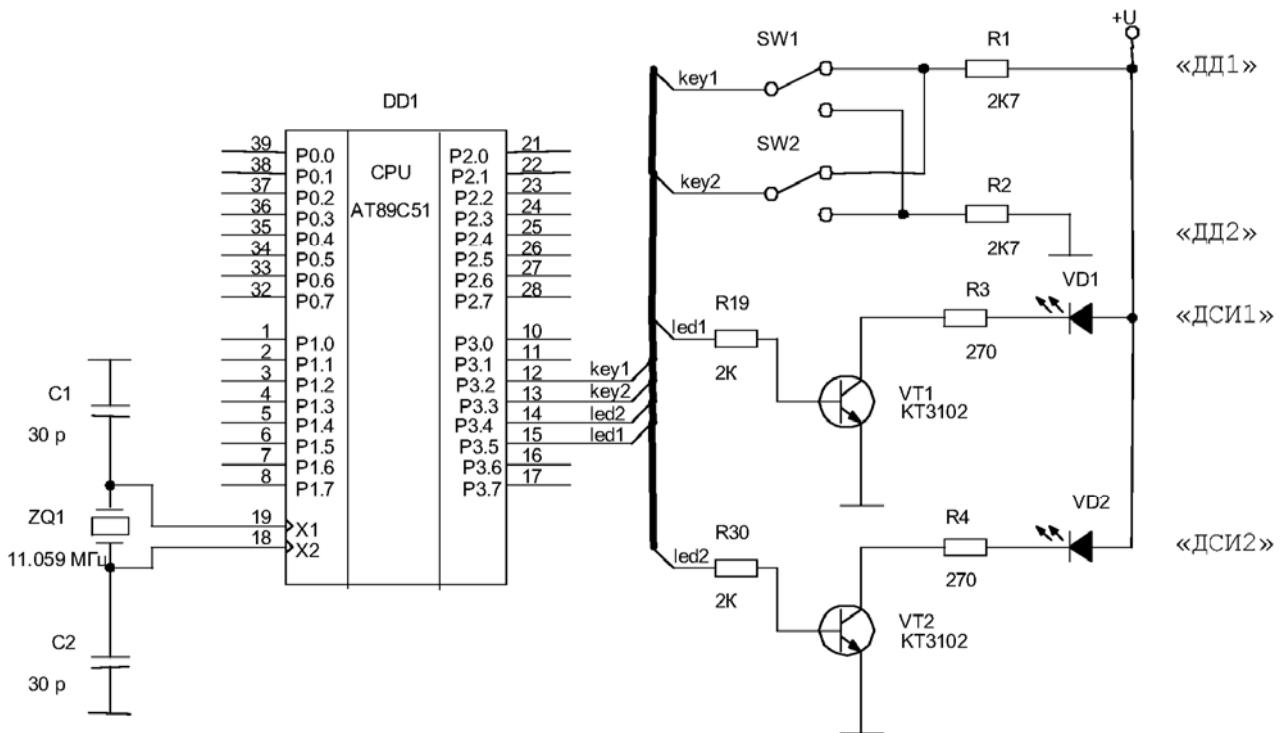


Рисунок 6.1 – Фрагмент схемы ввода-вывода дискретных сигналов

На рисунке 6.2 показана блок-схема алгоритма.

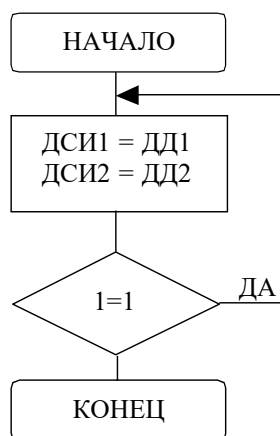


Рисунок 6.2 – Блок-схема алгоритма

Листинг программы для решения задачи

```
#include <reg51.h>
```

```
sbit DD1 = P3^2;
sbit DD2 = P3^3;
sbit DSI1 = P3^5;
sbit DSI2 = P3^4;
```

```
void main (void)
{
```

```

while (1)
{
DSI1 = DD1;
DSI2 = DD2;
}
}

```

Работа с портами в микроконтроллерах AVR организована несколько сложнее. Обращение к портам производится через регистры ввода/вывода. Под каждый порт в адресном пространстве ввода/вывода зарезервировано по три адреса. По этим адресам размещаются три регистра: регистр данных порта PORTx, регистр направления данных DDRx и регистр выводов порта PINx. Под «x» подразумевается конкретная буква – имя порта – например, PORTA, DDRA и PINA. Например, формат регистров управления контроллеров ATtiny 11x/12x приведен на рисунке 6.3.

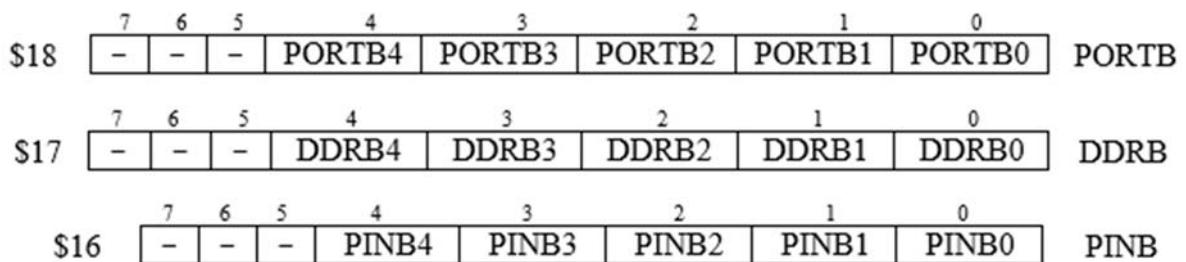


Рисунок 6.3 – Формат регистров PORTB, DDRB и PINB в моделях ATtiny 11x/12x

Любой порт ввода/вывода микроконтроллера серии AVR устроен таким образом, что каждый его разряд может работать как на ввод, так и на вывод. То есть он может быть входом, а может быть выходом. Направление передачи данных определяется содержимым регистра направления данных DDRB. Если разряд DDRB_n этого регистра установлен в «1», соответствующий n-й вывод порта является выходом, если сброшен в «0» – входом.

Для того чтобы выдать информацию на внешний вывод микросхемы, нужно в соответствующий разряд DDRB записать логическую единицу, а затем записать байт данных в регистр PORTB. Для того чтобы прочитать информацию с внешнего вывода контроллера, нужно записать в соответствующий разряд регистра DDRB «0», затем необходимо прочесть байт из регистра PINB.

Порты ввода/вывода микроконтроллеров AVR имеют еще одну полезную функцию. В режиме ввода информации они могут при необходимости подключать к каждому выводу порта внутренний нагрузочный резистор. Внутренний резистор позволяет значительно расширить возможности порта. Такой резистор создает вытекающий ток для внешних устройств, подключенных между выводом порта и общим проводом.

Программа, реализующая задачу, аналогичную приведенной выше, с учетом

подключения SW1 и SW2 к линиям PG. 0 и PG.1, а VD1 и VD2 к линиям – PG.4 и PG.3 может выглядеть как

```
#include <iom128.h>

void main (void)
{
/* начальные установки порт G */
DDRG |= ( 1 << DDG3 ); /* Линия PORTG4 настроена как выход */
DDRG |= ( 1 << DDG4 ); /* Линия PORTG4 настроена как выход */while(
1 )
{
if ( PING & ( 1 << PG0 ) ) PORTG
&= ~( 1 << PORTG4 );
else
PORTG |= ( 1 << PORTG4 );if (
PING & ( 1 << PG1 ) )
PORTG &= ~( 1 << PORTG3 );
else
PORTG |= ( 1 << PORTG3 );
}
}
```

Более подробные сведения о разработке программ обработки дискретных сигналов можно найти в методических указаниях к практическому занятию № 6, электронная версия которых имеется в классе ПЭВМ кафедры (лаб. 207/2) по следующему адресу: D:\методические рекомендации\СТАНДАРТЫ РФ\Специальность 15.03.06\Микропроцессорная техника в мехатронике и робототехнике \МПТ ПЗ 06.pdf.

7 Практическое занятие № 7. Составление алгоритмов и программ, реализующих ввод и обработку аналоговых сигналов

Цель занятия: изучить методику разработки программ обработки аналоговых сигналов.

Задание

По заданной преподавателем схеме составить блок-схему алгоритма, а также программу, производящую обработку информации.

При вводе аналоговых сигналов может понадобиться решить следующие задачи:

- отфильтровать;
- усилить до уровня преобразования АЦП;
- гальванически развязать;

- преобразовать аналоговый сигнал в цифровой код.

При выводе:

- запомнить значение выходного кода;
- произвести гальваническую развязку;
- преобразовать цифровой код в аналоговый сигнал;
- усилить сигнал по уровню.

При разработке программы, обеспечивающей ввод-вывод аналоговых сигналов, необходимо учесть способ подключения АЦП/ЦАП в системе.

Первый – использование внешних преобразователей. Такой способ применяется в тех случаях, если в структуре используемого микроконтроллера АЦП/ЦАП не предусмотрен или все его возможности уже использованы. В этом случае преобразователь подключается к выводам контроллера и работа с ним осуществляется путем обмена цифровой информацией и выдачи дискретных сигналов, как было рассмотрено на предыдущем занятии. Выдачу сигналов управления на выводы АЦП/ЦАП необходимо осуществлять в строгом соответствии с временными диаграммами, представленными в документации преобразователя.

Второй – наличие в структуре контроллера АЦП/ЦАП, возможностей которого достаточно для решения задач ввода-вывода информации. В этом случае работа с преобразователем осуществляется при помощи внутренних регистров управления микроконтроллера и все, что потребуется разработчику программы, – обратиться к соответствующим регистрам, прочитать хранящуюся или записать новую информацию.

Так, большинство микроконтроллеров AVR, например, ATMega16, содержит в себе 10-битовый АЦП, вход которого может быть соединён с одним из восьми выводов PortA. АЦП Mega16, как и любому другому АЦП, нужно опорное напряжение для целей сравнения со входным (если измеряемое равно опорному, то получаем максимальный код в двоичном виде). Опорное напряжение подаётся на вывод ADRef или может использоваться внутренний генератор с фиксированным напряжением 2,65 В.

АЦП включается установкой бита ADEN в регистре ADCSRA. После преобразования 10-битный результат оказывается в 8-битных регистрах ADCL и ADCH. По умолчанию младший бит результата находится справа (то есть в bit 0 регистра ADCL, так называемое правое выравнивание). Но порядок следования битов на левое выравнивание можно сменить, установив бит ADLAR в регистре ADMUX. Это удобно, если требуется получить 8-битовый результат. В таком случае требуется прочитать только регистр ADCH. В противном случае Вы должны сначала прочитать регистр ADCL первым, а ADCH вторым, чтобы быть уверенным в том, что чтение этих двух регистров относится к результату одного преобразования.

Одиночное преобразование может быть вызвано записью бита ADSC в регистр ADCSRA. Этот бит остаётся установленным всё время, занимаемое преобразованием. Когда преобразование закончено, бит автоматически устанавливается в 0. Можно также начинать преобразования по событиям из разных источников. Модуль АЦП также может работать в режиме непрерывного

преобразования. В таком случае АЦП постоянно производит преобразование и обновляет регистры ADCH и ADCL новыми значениями.

Для выполнения преобразования модулю АЦП необходима тактовая частота. Чем выше эта частота, тем быстрее будет происходить преобразование (оно обычно занимает 13 тактов, первое преобразование занимает 25 тактов). Но чем выше частота (и выше скорость преобразования), тем менее точным получается результат. Для получения максимально точного результата модуль АЦП должен тактироваться частотой в пределах от 50 до 200 кГц. Если необходим результат с точностью менее 10 бит, то можно использовать частоту больше 200 кГц. Модуль АЦП содержит делитель частоты, чтобы получать нужную тактовую частоту для преобразования из частоты процессора.

Регистр ADMUX задаёт входной контакт порта А для подключения АЦП, ориентирование результата и выбор опорной частоты. Если установлен бит ADLAR, то результат выравнивается по левому краю. Биты REFS0-REFS1 отвечают за выбор опорного напряжения (рисунок 7.1).

Bit	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 7.1 – Формат регистра ADMUX контролера ATmega16

Биты REFS0 – REFS1 отвечают за выбор опорного напряжения (таблица 7.1).

Таблица 7.1

REFS1	REFS0	Voltage Reference Selection
0	0	AREF. Внутренний ИОН отключён
0	1	AVCC с внешним конденсатором на ножке AREF
1	0	Зарезервировано
1	1	Внутренний ИОН на 2,56 В, с внешним конденсатором на AREF

Выбор контакта ввода выполняется при помощи поля MUX[4:0]. В таблице 7.2 приведен выбор контакта для однопроводного подключения.

Регистр управления и статуса АЦП ADCSRA (рисунок 7.2) используется для управления работой АЦП.

Бит ADEN = 1 включает модуль АЦП.

Запись единицы в ADSC запускает цикл преобразования. В режиме непрерывного преобразования запись единицы запускает первое преобразование, последующие запускаются автоматически.

Таблица 7.2

MUX4...0	Однопроводный вход
00000	ADC0
00001	ADC1
00010	ADC2
00011	ADC3
00100	ADC4
00101	ADC5
00110	ADC6
00111	ADC7

Bit	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	RW	RW	RW	RW	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 7.2 – Формат регистра ADCSRA контроллера ATmega16

ADIF – флаг прерывания АЦП. Этот бит устанавливается в 1, когда АЦП завершено преобразование и в регистрах ADCL и ADCH находятся актуальные данные. Флаг устанавливается даже в том случае, если прерывания запрещены. Это необходимо для случая программного опроса АЦП; если прерывания, то флаг сбрасывается автоматически; если программный опрос – флаг может быть сброшен записью лог.1 в этот бит.

ADIE – если в этом бите установлена 1, и прерывания разрешены глобально, то при окончании преобразования будет выполнен переход по вектору прерывания от АЦП.

Биты ADPS[2:0] (таблица 7.3) задают коэффициенты делителя частоты.

Таблица 7.3

ADPS2	ADPS1	ADPS0	Коэффициент деления
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	62
1	1	1	128

Таким образом, для того чтобы использовать АЦП контроллера AVR, его необходимо предварительно настроить:

- включить АЦП;

- выбрать источник опорного напряжения;
- выбрать вывод, для которого необходимо произвести преобразование;
- настроить частоту тактового сигнала;
- выбрать способ выравнивания результата преобразования;
- выбрать режим преобразования.

После этого преобразование можно запустить и, дождавшись окончания процесса, прочитать результат из регистров – сначала из ADCL, затем из ADCH, содержимое которых формируют 10-разрядный результат.

Более подробные сведения о разработке программ обработки аналоговой информации можно найти в методических указаниях к практическому занятию № 7, электронная версия которых имеется в классе ПЭВМ кафедры (лаб. 207/2) по следующему адресу: D:\методические рекомендации\СТАНДАРТЫ РФ\Специальность 15.03.06\Микропроцессорная техника в мехатронике и робототехнике \МПТ ПЗ 07.pdf.

8 Практическое занятие № 8. Составление алгоритмов и программ, реализующих временные функции управления

Цель занятия: изучить методику разработки программ, реализующих временные функции.

Задание

По заданной преподавателем схеме составить блок-схему алгоритма, а также программу, производящую обработку информации.

Микропроцессорные системы управления являются системами, работающими в режиме реального времени, подразумевающим формирование сигналов управления в определенные моменты времени, реакцию на появляющиеся события. При реализации временных функций в микропроцессорных системах можно воспользоваться одним из следующих способов:

- программным – с использованием исключительно программных средств микропроцессора;
- аппаратным – с использованием исключительно аппаратных средств системы;
- программно-аппаратным – с использованием как аппаратных, так и программных средств системы.

Любая команда микропроцессором выполняется за определенное время. При программном способе разрабатывается программа, набор команд в которой подобран таким образом, чтобы программа выполнялась в течение заданного промежутка времени.

При аппаратном способе для формирования задержек времени используются аппаратные элементы (таймеры, счетчики) и система прерываний микропроцессора, позволяющие во время выполнения программы осуществлять выдержку времени, а по ее окончании выполнить необходимые действия в

подпрограмме обработки прерываний.

Программно-аппаратный способ основывается на использовании аппаратных элементов для выдержки времени и программном контроле их работы. После окончания выдержки времени аппаратные элементы устанавливают аппаратные флаги, опрашивая которые процессор может судить об окончании процесса.

При использовании программного способа процедура реализации временной задержки может использовать метод программных циклов. При этом в некоторый рабочий регистр загружается число X , которое затем в каждом проходе цикла уменьшается на 1. Так продолжается до тех пор, пока содержимое рабочего регистра не станет равным нулю, что интерпретируется программой как момент выхода из цикла. Время задержки при этом определяется числом, загруженным в рабочий регистр, и временем выполнения команд, образующих программный цикл. Поскольку в программе может потребоваться многократно формировать временные задержки, фрагмент программы, реализующей задержку, целесообразно оформить в виде подпрограммы. Алгоритм, реализующий такую программу, может выглядеть, как на рисунке 8.1.

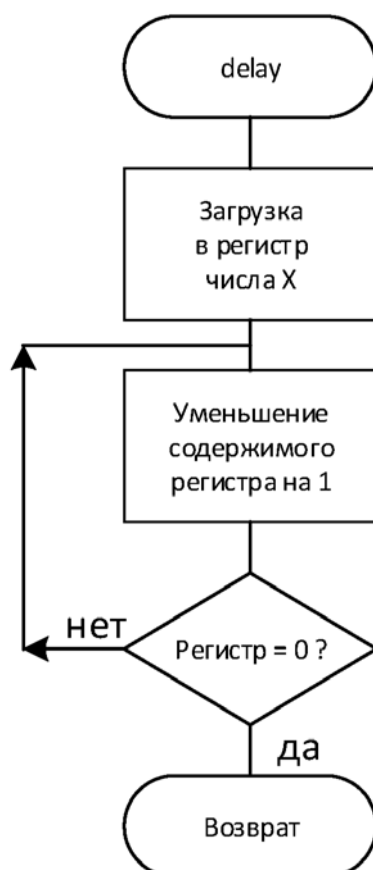


Рисунок 8.1 – Блок-схема подпрограммы задержки времени

Для получения требуемой временной задержки необходимо определить число X , загружаемое в рабочий регистр. Определение числа X выполняется на

основе расчёта времени выполнения команд, образующих данную подпрограмму. При этом необходимо учитывать, что команды вызова и возврата из подпрограммы выполняются однократно, а число повторений команд декремента регистра и условного перехода равно числу X .

Общая задержка времени данного алгоритма может быть вычислена с помощью выражения

$$T = (N_{\text{вызов}} + N_{\text{загрузки_регистра}} + X \cdot (N_{\text{декремент}} + N_{\text{условного_перехода}}) + N_{\text{возврат}}) \cdot T_{\text{Ц}}, \quad (8.1)$$

где N – количество машинных циклов команд отдельных блоков;

$T_{\text{Ц}}$ – длительность машинного цикла процессора.

Таким образом, чтобы получить исходные данные для расчета числа X , необходимо составить программу, реализующую приведенный алгоритм, и найти время выполнения.

Например, для процессора семейства MCS-51 программу можно составить с использованием следующих команд:

- CALL rel_addr – вызов подпрограммы – 2 МЦ;
- MOV Ri, #data – загрузка регистра числом data – 1 МЦ;
- DJNZ Ri, rel_addr – декремент регистра и переход, если результат ненулевой – 2 МЦ;
- RET – возврат из подпрограммы – 2 МЦ.

Длительность машинного цикла процессора – 12 тактов тактовой частоты.

Для процессора AVR можно использовать команды:

- RCALL k – вызов подпрограммы – 3 МЦ;
- LDI Ri, #data – загрузка регистра числом – 1 МЦ;
- DEC Ri – декремент регистра – 1 МЦ;
- BRNE k – переход по адресу, если не нуль, – 1 МЦ;
- RET – возврат из подпрограммы – 4 МЦ.

Длительность машинного цикла процессора равна длительности такта тактовой частоты. При этом общая производительность процессора равна 1MIPS/1МГц, а тактовая частота контроллера ATmega 16 может достигать 16 МГц.

Для процессора ARM процессора семейства Cortex-M4 можно использовать команды:

- BL link – переход по метке с сохранением текущего значения счетчика команд в регистре связи LR – 3 МЦ;
- LDR Ri, = data – загрузка регистра числом – 2 МЦ;
- SUBS Ri, #data – уменьшение регистра на значение data с обновлением флагов процессора – 1 МЦ;
- BNE link – переход по метке, если не нуль, – 2 МЦ;
- BX LR – переход по адресу, находящемуся в регистре связи, – 3 МЦ.

Длительность машинного цикла процессора равна длительности такта частоты системной шины АНВ, значение которой, например, для контроллера STM32F407 может достигать 168 МГц.

Таким образом, задержка времени, реализуемая данным способом, зависит

от тактовой частоты процессора и числа, загружаемого в регистр. Следует помнить, что разрядность регистров общего назначения микроконтроллера Cortex-M4 равна 32, а для микроконтроллеров MCS-51 и AVR – 8. В связи с этим общая задержка для микроконтроллеров MCS-51 и AVR будет составлять доли миллисекунды. Задержка большой длительности может быть реализована методом вложенных циклов. Суть метода аналогична ранее рассмотренному, однако для выполнения одного из циклов используется один регистр, а для вложенного в него – другой.

Недостатком программного способа реализации временной задержки является нерациональное использование ресурсов микроконтроллера: во время формирования задержки он практически простаивает, т. к. не может решать никаких задач управления объектом. В то же время аппаратные средства позволяют реализовать временные задержки на фоне основной программы работы.

Для реализации временных функций аппаратными средствами необходимо использовать специальные устройства, позволяющие обеспечить выдержки времени. Эту задачу можно решить с использованием таймеров-счетчиков. Большинство современных однокристальных микроконтроллеров оснащены встроенными таймерами. Кроме того, с целью расширения возможностей контроллера можно использовать внешние микросхемы таймеров, выполняющие схожие функции.

Так, например, базовая конфигурация микроконтроллеров MCS-51 (например, микросхема AT89C51) содержит два 16-разрядных суммирующих таймера/счетчика TC0 и TC1, предназначенных для подсчета внешних событий, получения программно-управляемых временных задержек и выполнения внутренних времязадающих функций.

Таймеры/счетчики (T/C) построены на программно-доступных регистровых парах (TH0, TL0) и (TH1, TL1), размещенных в области регистров специальных функций SFR. В управлении режимом работы TC0 и TC1 участвует старшая тетрада регистра TCON и регистр TMOD. Регистр TMOD отвечает за настройку режимов работы T/C. На рисунке 8.2 приведен формат регистра, а в таблицах 8.1 и 8.2 – назначение отдельных разрядов.

GATE	C/T	M1	M0	GATE	C/T	M1	M0
настройка канала 1 T/C				настройка канала 0 T/C			

Рисунок 8.2 – Регистр режимов работы таймера/счетчика TMOD

Младшая тетрада регистра TCOM, формат которого приведён на рисунке 8.3, используется в работе системы прерываний.

Старшая тетрада регистра TCON, назначение разрядов которой приведено в таблице 8.3, содержит биты разрешения счета TRx.

Оба регистра управления расположены в области SFR и программно доступны, а регистр TCON, находящийся ещё и в области BSEG, имеет программно-доступные биты.

При работе в качестве таймера в любом из указанных выше режимов содержимое Т/С инкрементируется в каждом машинном цикле, т. е. через каждые 12 периодов внутреннего генератора. Отличие режимов работы заключается в разрядности используемых регистров и, следовательно, в максимальной формируемой выдержке времени.

Таблица 8.1

Символ	Разряд	Имя и назначение
GATE	TMOD.7 для Т/С1 TMOD.3 для Т/С0	Управление блокировкой. Если бит установлен, то таймер/счётчик «х» разрешён до тех пор, пока на входе «INT х» высокий уровень и бит управления «TRх» установлен. Если бит сброшен, то Т/С разрешается, как только бит управления «TRх» устанавливается
G/T	TMOD.6 для Т/С1 TMOD.2 для Т/С0	Бит выбора таймера или счётчика событий. Если бит сброшен, то работает таймер от внутреннего источника сигналов синхронизации. Если бит установлен, то работает счётчик от внешних сигналов на входе «Тх»
M1	TMOD.5 для Т/С1 TMOD.1 для Т/С0	Режим работы (см. таблицу 8.2)
M0	TMOD.4 для Т/С1 TMOD.0 для Т/С0	

Таблица 8.2

M1	M0	Режим работы
0	0	TL работает как 5-битный предделитель
0	1	16-битный таймер/счётчик ТН хранит значение, которое должно быть перезагружено в TL каждый раз по переполнению
1	0	Таймер/счётчик 1 останавливается. Таймер/счётчик 0: TL0 работает как 8-битный таймер/счётчик и его режим определяется управляющими битами таймера 0. ТНО работает только как 8-битный таймер, и его режим определяется управляющими битами таймера 1
1	1	

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
-----	-----	-----	-----	-----	-----	-----	-----

Рисунок 8.3 – Регистр статуса и управления TCON

Так, в режиме 0 значение имеют только младшие 5 бит регистра TLx и 8 разрядов ТНх. Переполнение TLx приводит к инкрементированию регистра ТНх. Процесс проходит до переполнения ТНх, в результате чего устанавливается флаг переполнения TFx. То есть в данном режиме регистры ТНх и ТНх образуют 13-битную регистровую пару, и, следовательно, таймер может подсчитать до 8192 импульсов, тактирующих таймер.

Режим 1 отличается от режима 0 тем, что используются все 8 бит регистра TLx, и таймер имеет разрядность 16, в результате чего может подсчитать 65536 импульсов.

В режиме 2 каждый Т/С может работать как автоперезагружаемый 8-разрядный таймер/счётчик (то есть может подсчитать 256 импульсов). В качестве счётчика используется регистр TLx, а THx содержит программно установленное начальное значение, с которого ведётся счёт. Перегрузка оставляет содержимое THx неизменным. При каждом очередном переполнении TLx устанавливается флаг переполнения TFx и автоматически начальное значение перезагружается из THx в TLx.

Таблица 8.3 – Назначение битов регистра TCON

Символ	Разряд	Имя и значение
TF1	TCON.7	Флаг переполнения таймера 1. Устанавливается аппаратно при переполнении таймера/счётчика. Сбрасывается при обслуживании прерывания аппаратно
TR1	TCON.6	Бит управления таймера 1. Устанавливается/сбрасывается программой для пуска/останова
TF0	TCON.5	Флаг переполнения таймера 0. Устанавливается аппаратно. Сбрасывается. Устанавливается аппаратно. Сбрасывается при обслуживании прерывания
TR0	TCON.4	Бит управления таймера 0. Устанавливается/сбрасывается программой для пуска/останова таймера/счётчика
IE1	TCON.3	Флаг фронта прерывания 1. Устанавливается аппаратно, когда детектируется срез внешнего сигнала INT1. Сбрасывается при обслуживании прерывания
IT1	TCON.2	Бит управления типом прерывания 1. Устанавливается/сбрасывается программно для спецификации запроса INT1 (срез / низкий уровень)
IE0	TCON.1	Флаг фронта прерывания 0. Устанавливается по срезу сигнала INT0. Сбрасывается при обслуживании прерывания
IT0	TCON.0	Бит управления типом прерывания 0. Устанавливается/сбрасывается программно для спецификации запроса INT0 (срез / низкий уровень)

Общая величина времени, которое пройдет с момента запуска таймера до его переполнения, будет составлять:

$$T = \frac{12}{f_{cpu}}(N_{max} - N_{THL}), \quad (8.2)$$

где f_{cpu} – тактовая частота процессора;

N_{max} – максимальное значение таймера, зависящее от разрядности, определяемой режимом работы;

N_{THL} – начальное значение, загружаемое в регистры THx, TLx.

Кроме того, следует иметь в виду, что, если использовать Т/С в режиме счетчика, можно обеспечить его тактирование импульса произвольной частоты и тем самым получить еще большую гибкость при формировании задержек времени.

Таким образом, для организации временных задержек с использованием Т/С микроконтроллеров MCS-51 можно воспользоваться следующей программно-аппаратной последовательностью действий:

- настроить режим работы предполагаемого к использованию таймера путем записи соответствующего управляющего слова в регистр TMOD;
- настроить требуемую задержку времени путем записи начального значения (с которого таймер начнёт счет) в регистры THx и TLx (в зависимости от режима работы);
- запустить таймер в работу путем установки бита запуска TRx;
- дождаться окончания подсчета, контролируя значение соответствующего флага переполнения;
- выполнить необходимые по окончании выдержки времени действия, после чего процедуру можно повторить.

Более подробные сведения о реализации временных функций можно найти в методических указаниях к практическому занятию № 8, электронная версия которых имеется в классе ПЭВМ кафедры (лаб. 207/2) по следующему адресу: D:\методические рекомендации\СТАНДАРТЫ РФ\Специальность 15.03.06\Микропроцессорная техника в мехатронике и робототехнике \МПТ ПЗ 08.pdf.

9 Практическое занятие № 9. Составление алгоритмов и программ, реализующих типовые функции обработки информации

Цель занятия: изучить методику разработки прикладного программного обеспечения.

Задание

По заданной преподавателем схеме составить блок-схему алгоритма, а также программу, производящую обработку информации.

Типовые прикладные программы в микропроцессорных системах требуют решения достаточно широкого круга задач:

- ввод-вывод информации (в цифровом и аналоговом виде);
- реализации временных функций;
- обработка информации по заданному закону;
- сохранения промежуточных результатов обработки информации и т. д.

Часто для их решения необходимо использовать внутренние и внешние элементы микропроцессорной системы – периферийные устройства и память.

При разработке микроконтроллерных систем могут быть использованы два способа организации прикладных программ: монолитный и модульный. При первом способе вся прикладная программа разрабатывается как единое целое. При втором она строится из отдельных программных блоков, каждый из которых реализует некоторую процедуру обработки данных или управления. Взаимосвязь блоков определяется разработчиком при монтаже из этих блоков законченной прикладной программы.

Отдельные фрагменты прикладной программы могут быть получены в виде линейной последовательности блоков, другие (многократно используемые) обычно оформляются в виде программ, к которым прикладная программа, называемая основной, имеет возможность обратиться по мере необходимости. Программа должна обладать следующими свойствами: выполнять законченную процедуру обработки данных, иметь только один вход и один выход и не обладать эффектом последействия, при котором текущее выполнение подпрограммы оказывало бы влияние на её последующие выполнения.

Если задача на разработку прикладной программы для микроконтроллера поставлена, то для получения текста исходной программы необходимо выполнить ряд последовательных действий.

- 1 Подробно описать задачу.
- 2 Проанализировать задачу.
- 3 Выполнить инженерную интерпретацию задачи, желательно с привлечением того или иного аппарата формализации (граф автоматов, сети Петри, матрицы состояний и т. п.).
- 4 Разработать общую схему алгоритма работы контроллера.
- 5 Разработать детализированные схемы отдельных процедур, выделенных на основе модульного принципа составления программ.
- 6 Детально проработать интерфейс контроллера и внести исправления в общую и детализированные схемы алгоритмов.
- 7 Распределить рабочие регистры и память.
- 8 Сформировать текст исходной программы.
- 9 Определить, что должен делать модуль.
- 10 Определить способы получения модулем исходных данных (от датчиков через порты ввода, из таблиц в памяти или через рабочие регистры).
- 11 Определить необходимость какой-либо предварительной обработки введенных исходных данных (маскирование, сдвиг, масштабирование, перекодировка).
- 12 Определить метод преобразования входных данных в требуемые выходные. Используя операторы процедур и условные операторы принятия решения, отобразить на языке схемы алгоритма выбранный метод.
- 13 Определить способы выдачи из модуля обработанных данных (передать в память, в вызывавшую программу или в порты вывода).
- 14 Определить необходимость какой-либо вторичной обработки выводимых данных (изменение формата, перекодирование, масштабирование, маскирование).
- 15 Вернуться к п. 4 перечня и проанализировать полученный результат. Выполнить итеративную корректировку схемы алгоритма с целью сделать её простой, логичной, стройной и обладающей четким графическим образом.
- 16 Проверить работоспособность алгоритма на бумаге путем подстановки в него действительных данных. Убедиться в его сходимости и результативности.
- 17 Рассмотреть предельные случаи и попытаться определить граничные значения информационных объектов, с которыми оперирует алгоритм, за пределами которых он теряет свойства конечности, сходимости и

результативности. Особое внимание при этом следует уделить анализу возможных ситуаций переполнения разрядной сетки, изменение знака результата операции, деление на переменную, которая может принять нулевое значение.

18 Провести мысленный эксперимент по определению работоспособности алгоритма в реальном масштабе времени, когда стохастические события, происходящие в объекте управления, могут оказать влияние на работу алгоритма. При этом самому тщательному анализу следует подвергнуть реакцию алгоритма на возможные прерывания с целью определения критических операторов, которые необходимо защитить от прерываний. Кроме того, в ходе этого мысленного эксперимента следует проанализировать логику алгоритма с целью определения таких последовательностей операторов, при выполнении которых микроконтроллер может «не заметить» кратковременных событий в объекте управления. При обнаружении таких ситуаций в логику следует внести коррективы. Более подробные сведения о принципах разработки типовых программ обработки информации можно найти в методических указаниях к практическому занятию № 9, электронная версия которых имеется в классе ПЭВМ кафедры (лаб. 207/2) по следующему адресу: D:\методические рекомендации\СТАНДАРТЫ РФ\Специальность 15.03.06\Микропроцессорная техника в мехатронике и робототехнике \МПТ ПЗ 09.pdf.

Список литературы

1 Микропроцессорная техника в мехатронике и робототехнике: методические рекомендации к лабораторным работам для студентов направления подготовки 15.03.06 «Мехатроника и робототехника» / сост. В. Н. Ситников. – Могилев: Беларус.-Рос. ун-т, 2018. – 42 с.

2 **Гуров, В. В.** Микропроцессорные системы: учебное пособие / В. В. Гуров. – Москва: ИНФРА-М, 2022. – 336 с.

3 **Жежера, Н. И.** Микропроцессорные системы автоматизации технологических процессов: учебное пособие / Н. И. Жежера. – 2-е изд. – Москва; Вологда: Инфра-Инженерия, 2020. – 240 с.

4 **Беккер, В. Ф.** Технические средства автоматизации. Интерфейсные устройства и микропроцессорные средства: учебное пособие / В. Ф. Беккер. – 2-е изд. – Москва: РИОР; ИНФРА-М, 2020. – 152 с.

5 **Гарелина, С. А.** Автоматизация измерений, испытаний и контроля: учебное пособие / С. А. Гарелина, К. П. Латышенко, И. Ю. Сергеев. – Железногорск: Сибирская пожарно-спасательная академия ГПС МЧС России, 2020. – 486 с.