

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

А.И. Солдатов, П.В. Сорокин,
А.А. Солдатов, Е.Ю. Киселева

ИССЛЕДОВАНИЕ ЭЛЕКТРОННЫХ ИНДИКАТОРОВ. ПРАКТИКУМ

*Рекомендовано в качестве учебного пособия
Редакционно-издательским советом
Томского политехнического университета*

2-е издание

Издательство
Томского политехнического университета
2014

УДК 619.615.83
ББК 22.32
С60

Солдатов А.И.

С60 Исследование электронных индикаторов. Практикум: учебное пособие / А.И. Солдатов, П.В. Сорокин, А.А. Солдатов, Е.Ю. Киселева, Томский политехнический университет. – 2-е изд. – Томск: Изд-во Томского политехнического университета, 2014. – 145 с.

В учебно-методическом пособии приведено описание шести лабораторных работ, охватывающих материал по изучению режимов работы, методов адресации и схем управления электронными индикаторами.

Содержание учебно-методического пособия соответствует курсу лабораторных работ, который авторы обеспечивают в Томском политехническом университете на кафедре «Промышленной и медицинской электроники» и предназначено для студентов высших технических учебных заведений, обучающихся по направлению подготовки специалистов 210100 «Электроника и нанoeлектроника» и 201000 «Биотехнические системы и технологии»

УДК 619.615.83
ББК 22.32

Рекомендовано к печати редакционно-издательским отделом
Томского политехнического университета

Рецензенты:

Доцент кафедры ПМЭ ТПУ, к.т.н.
А.Ф. Глотов

Зав. каф. БМК СибГМУ Томск, к.т.н., профессор
Я.С. Пеккер

Доцент каф. СЭ ТУСУР, к.т.н.
М.Е. Антипин

© Томский политехнический университет, 2014
© Солдатов А.И., Макаров В.С., Сорокин П.В., 2009
© Оформление. Издательство ТПУ, 2014

Содержание:

ВВЕДЕНИЕ.....	4
ЛАБОРАТОРНАЯ РАБОТА №1. Исследование режимов работы дискретных полупроводниковых Индикаторов.....	5
ЛАБОРАТОРНАЯ РАБОТА №2. Изучение принципов формирования изображения на полупроводниковых многоразрядных семисегментных индикаторах.....	17
ЛАБОРАТОРНАЯ РАБОТА №3. Изучение принципов формирования изображения на жидкокристаллических многоразрядных семисегментных индикаторах.....	36
ЛАБОРАТОРНАЯ РАБОТА №4. Исследование способов формирования изображения на матричных Экранах.....	47
ЛАБОРАТОРНАЯ РАБОТА №5. Изучение принципов формирования изображения на электронно — лучевых трубках.....	65
ЛАБОРАТОРНАЯ РАБОТА №6. Исследование матричного жидкокристаллического индикатора с встроенной схемой управления.....	74
ПРИЛОЖЕНИЕ А.....	92
ПРИЛОЖЕНИЕ Б.....	120

ВВЕДЕНИЕ

Учебно-методическое пособие предназначено для самостоятельной подготовке студентов направлений 201000 «Биотехнические системы и технологии» и 210100 «Электроника и наноэлектроника» к проведению лабораторных работ по курсу «Системы обработки и отображения информации». Учебно-методическое пособие содержит методические рекомендации к проведению 6 лабораторных работ и два приложения, содержащих описание средств программирования и отладки микроконтроллера с8051 и основы программирования на языке СИ.

Весь набор лабораторных работ охватывает основную часть теоретического курса и позволяет студентам получить практические навыки работы с электронными индикаторами, схемами их управления и написании программ управления дискретными индикаторами с помощью микроконтроллера.

Приложение А содержит описание средств программирования и отладки микроконтроллера с8051, на базе которого реализован лабораторный стенд. В пособии очень подробно показан ход создания проекта, написания и отладки программы, как на языке низкого уровня - Assembler, так и на языке высокого уровня СИ, записи ее в микроконтроллер и соединения лабораторного стенда с персональным компьютером. Приложение изобилует рисунками, позволяющими в наглядной форме получить информацию о процедуре создания проекта, написании программы, ее отладки и компиляции.

Приложение Б содержит основы программирования микроконтроллеров на языке СИ. В приложении приведены основные понятия языка СИ, описаны функции и примеры их практического использования. Применение языка высокого уровня для программирования микроконтроллера позволяет существенно сократить время написания программы и получить навыки работы с языком СИ.

ЛАБОРАТОРНАЯ РАБОТА №1

ИССЛЕДОВАНИЕ РЕЖИМОВ РАБОТЫ ДИСКРЕТНЫХ ПОЛУПРОВОДНИКОВЫХ ИНДИКАТОРОВ

Цель работы: ознакомление студентов с основными схемами включения дискретных полупроводниковых индикаторов в статическом и динамическом режимах работы.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Дискретные индикаторы это прибор, информационное поле которого состоит из фиксированных в пространстве отдельных элементов отображения (Э.О.), а изображение создается одним Э.О. или их совокупностью. Дискретные индикаторы работают в 2-х режимах:

1) статической индикации, когда состояние элемента отображения меняется только при обновлении воспроизводимой информации, причем все выбранные элементы отображения работают одновременно. Статическая индикация может быть непрерывной $q = 1$ (рис. 1.1.а) или импульсной $q > 1$ (рис. 1.1.б),

2) динамической индикации (рис.1.1.в), характеризуемой тем, что разные элементы или группы элементов отображения, образующие индикаторное поле, включаются в разные части периода кадра T_k . При этом, пока воспроизводимая информация остается неизменной, фаза и длительность включенного состояния элемента сохраняются, т. е. $T_{обн} > T_k$ (рис.1.1.г). Для получения немелькающего изображения должно выполняться условие :

$$f_k \geq f_{кчм} \quad (1),$$

где: $f_{кчм}$ - критическая частота мелькания,

f_k - частота кадра,

а за период кадра необходимо адресовать все элементы, составляющие изображение. Поэтому скважность оказывается пропорциональной числу групп элементов отображения.

Такой режим регенерации изображения характеризуется следующими соотношениями:

- для яркости :

$$L_{вкж} = L_{vu} / q_v \quad (2)$$

где: $L_{v\text{каж}}$ – кажущаяся (усредненная во времени) яркость;

$L_{\text{ви}}$ - мгновенная яркость;

q_v -скважность импульсов излучения.

Средняя (кажущаяся) яркость мелькающего элемента $L_{\text{каж}}$ воспринимаемая глазом, определяется законом Тальботта:

$$L_{\text{каж}} = \frac{1}{T} \cdot \int_{t_1}^{t_2} L(t) dt, \quad (1.22)$$

где $L(t)$ – закон изменения яркости в интервале времени ($t_2 - t_1$),

T – период повторения мерцания элемента.

- для времени выборки :

$$t_{\text{в}} = \frac{1}{f_{\text{к}} \cdot q} = \frac{T_{\text{к}}}{q}; \quad (3)$$

где: $f_{\text{к}}$ - частота кадра;

$T_{\text{к}}$ - период кадра;

q - количество элементов отображения;

из-за инерционности индикаторов обычно $q \leq q_v$.

Для управления дискретными индикаторами используются в основном четыре метода адресации:

1. однокоординатный;
2. двухкоординатный;
3. многоуровневый;
4. со сканированием.

В данной лабораторной работе используется однокоординатный метод адресации (рис.1.2).

Однокоординатный способ адресации заключается в том, что каждый элемент отображения имеет два независимых входа управления, к которым прикладывается сигнал со схемы возбуждения. Очевидно, что независимость элементов отображения друг от друга сохраняется и тогда, когда для уменьшения числа соединений их вторые управляющие входы объединяются.

Так как входы независимы, то при однокоординатном способе адресации элементы отображения могут включаться одновременно и на любой промежуток времени, что позволяет организовать статическую

индикацию, т. е. использовать умеренные значения яркостей L_{vi} и большие времена выборки t_B .

Преимуществом однокоординатного способа адресации является также отсутствие жестких требований к параметрам индикаторов.

В то же время схемы однокоординатной адресации многоэлементных индикаторов имеют существенные недостатки — большое число каналов управления

$$N_y = N_{\text{эо}}$$

и выводов индикатора

$$N_B \geq N_{\text{эо}} + 1$$

Например, для 7-сегментного индикатора с десятичной точкой на один разряд число выводов равно $8+1=9$, а на три разряда уже 27. Если взять буквенный индикатор, у которого знакоместо образовано матрицей точек 5×7 , то на три знакоместа уже понадобится $35 \times 3 + 1 = 106$ выводов. Из-за трудности создания индикаторов и схем управления с большим числом выводов и в особенности их соединения между собой применение методов с однокоординатной адресацией ограничивается цифровыми УОИ на три-четыре знакоместа или буквенными СОИ на одно знакоместо.

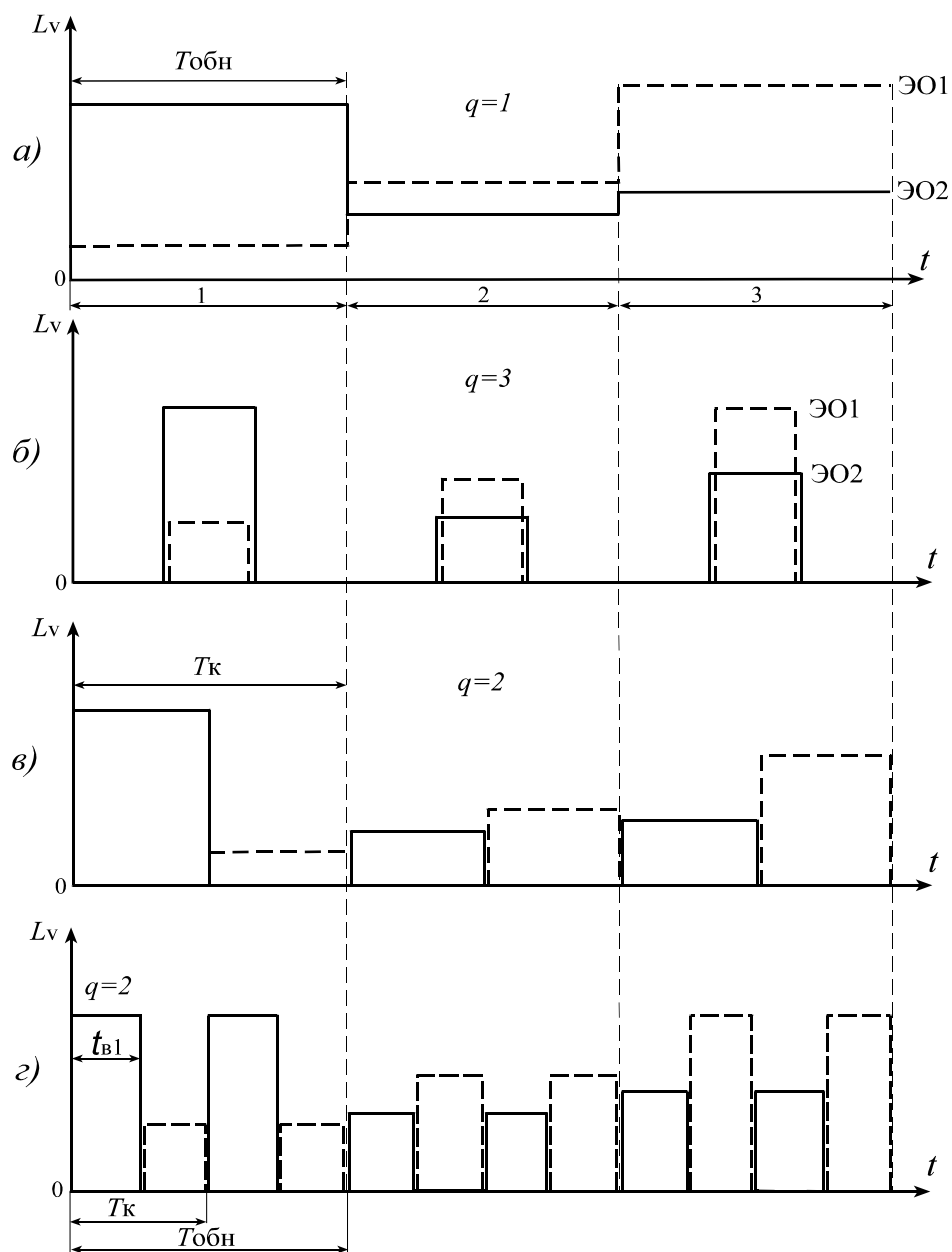


Рис. 1.1. Диаграммы работы дискретных индикаторов в различных режимах, а - статическом непрерывном, б – статическом импульсном, в и г - динамическом.

Принципиальная схема включения полупроводниковых индикаторов при однокоординатном методе адресации приведена на рис.1.3.

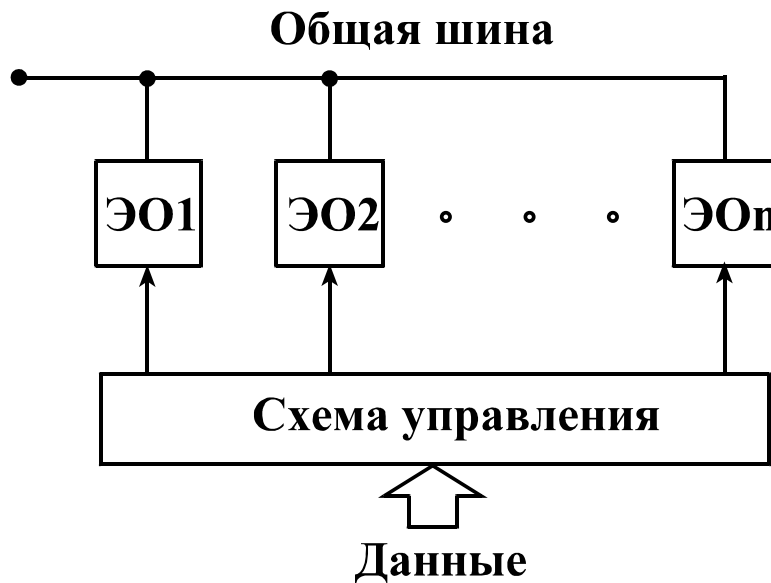


Рис.1.2. Структурная схема подключения элементов отображения при однокоординатном методе адресации.

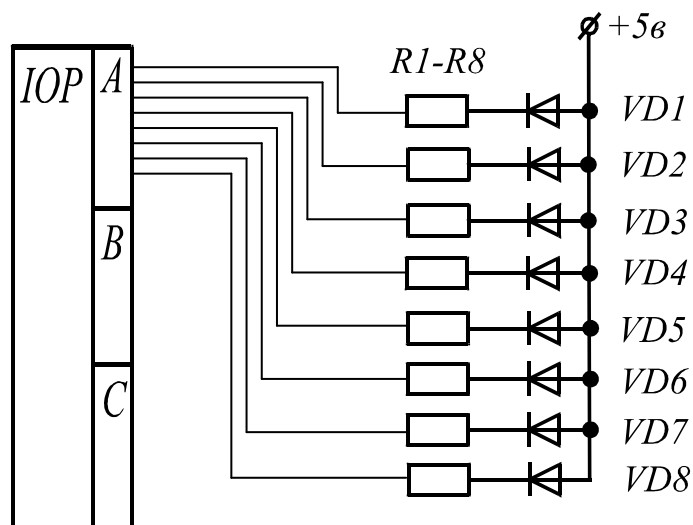


Рис.1.3. Принципиальная схема включения полупроводниковых индикаторов при однокоординатном методе адресации.

При работе в статическом импульсном или динамическом режиме для сохранения такого же значения кажущейся яркости свечения индикаторов как и в непрерывном режиме необходимо увеличивать мгновенное значение яркости $L_{\text{ви}}$.

Для обеспечения высокого значения $L_{\text{ви}}$ необходимо увеличивать импульсный ток через индикатор в соответствии с формулой :

$$I_{\text{имп}} = I_n * q^{1/n}$$

где I_n – постоянный прямой ток через индикатор, q – скважность, $n=1,2$ при $q \geq 5$ и $n=1,4$ при $q < 5$.

Импульсный ток задается резисторами $R1 - R8$, которые определяются из выражения:

$$R1 = \frac{E_n - U_{vd}}{I_{имп}} \quad (1),$$

где: E_n - напряжение питания,

U_{vd} - прямое напряжение на светодиоде,

$I_{имп}$ - импульсный ток светодиода.

ОПИСАНИЕ ЛАБОРАТОРНОГО СТЕНДА

В лабораторной работе используются светодиодные индикаторы типа L-934SR красного цвета. Их основные параметры приведены в таблице 1.1.

Управление полупроводниковыми индикаторами осуществляется через порты P5 и AD микроконтроллера C8051F060 рис 1.5. Аноды индикаторов VD1 – VD8 соединены вместе и подключены к плюсу источника питания. Катоды индикаторов через регистр памяти DD16 и буферный регистр DD13 подключены к порту P5 микроконтроллера DD10. Для записи данных в регистр памяти необходимо сформировать импульс записи CL4. Для этого на выходе порта AD микроконтроллера DD10 необходимо установить код 00000100b и по сигналу ALE он запишется в регистр памяти DD12. С выхода регистра этот код поступает на дешифратор DD23. По сигналу IWR с порта P3.5 на 4 выходе дешифратора появляется активный уровень, который разрешает запись данных в регистр DD16. Ток через индикаторы ограничивается резисторами R1 – R8.

Технические характеристики полупроводниковых индикаторов L-934S

Таблица 1.1

Основные параметры	L-934SR	L-934SB	L-934SG	L-934SY
Цвет свечения	красный	голубой	зеленый	желтый
Яркость свечения, мД/м ²	2-3,5	2-3,5	2-3,5	2-3,5
Напряжение, В: прямое	2	2	2	2
обратное	≤2	≤2	≤2	≤2
Ток, мА				
прямой номинальный	10	10	10	10
прямой максимальный	20	20	22	22

Пример алгоритма для включения светодиодов приведен на рис.1.4.:

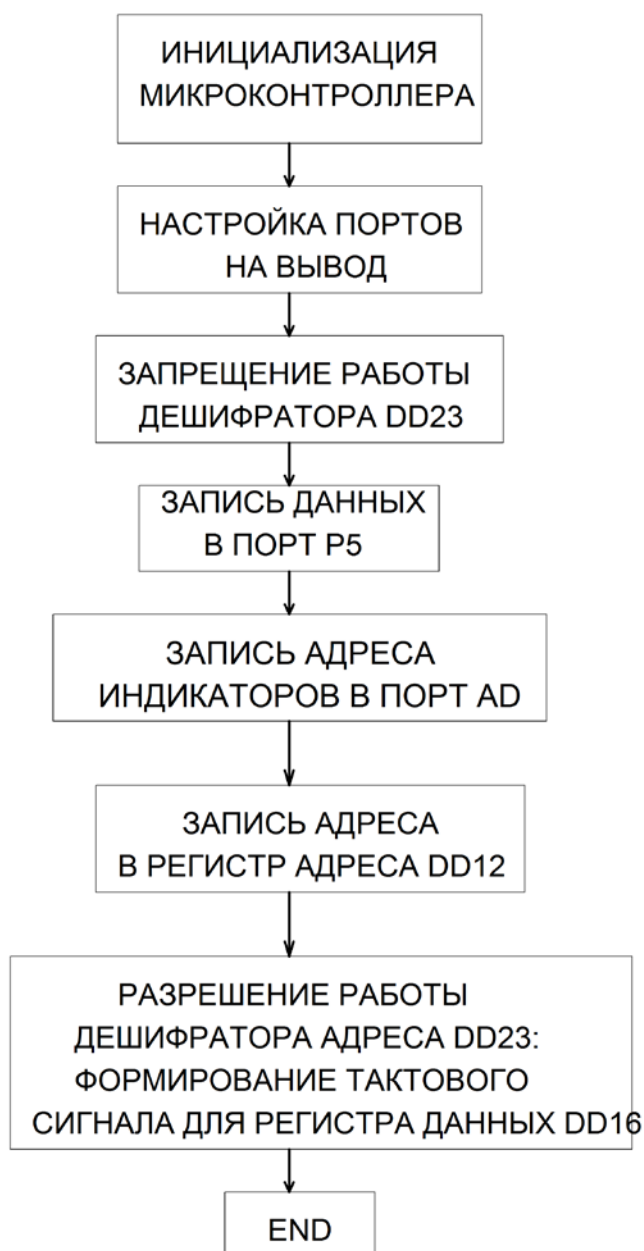


Рис.1.4.Алгоритм управления точечными полупроводниковыми индикаторами

По приведенному алгоритму разработаны программы включения 8 индикаторов:

- код программы на языке ассемблера.

`$INCLUDE (C8051F060.inc)`

`;-----`

`; настройка стека`

`;-----`

```

STACK          SEGMENT IDATA ;выбор сегмента стека
RSEG STACK
DS 80h ;резервирование 128 байтов для стека
;-----
; настройка таймеров
;-----
        CSEG AT 0
        ljmp  Main

Temp_2      SEGMENT CODE ;
RSEG Temp_2 ;выбор сегмента CODE
USING 0 ;использование нулевого банка регистров

Main:
        mov  WDTCN, #0deh ;выключение watchdog timer
        mov  WDTCN, #0adh
        mov  SP, #STACK-1 ;инициализация указателя стека

        mov  SFRPAGE, #0FH ;выбор страницы 0fH
        MOV  XBR2, #40H
        MOV  P3MDOUT, #0FFH ;
        MOV  P4MDOUT, #0FFH ;
        MOV  P5MDOUT, #0FFH ;
        MOV  P7MDOUT, #0FFH ;настройка портов P3-P5, P7 на
ВЫВОД
;-----
; основная программа
;-----
        SETB P3.5 ; запрет работы дешифратора
        CLR  P4.5 ;запрет работы регистра DD12
        MOV  P5, #0FH ;выбор 8 светодиодов
        MOV  P7, #04H ;включение регистра выбора индикатора
        SETB P4.5 ; Разрешение работы регистра DD12
        CLR  P3.5 ; Разрешение работы дешифратора
        RET
        END

```

- код программы на языке C++.

```

#include "C8051F060.h"
sbit a=P3^5;
sbit b=P4^5;

```

```

void PORT_sett (void);
void work (void);
void preobr (void);
int d=0x00,f=0xff;           // d-количество горящих светодиодов
                               // f-тушение светодиодов

void main (void)
{
    SFRPAGE = CONFIG_PAGE;
    EA=0;                      //
    WDTCN = 0xDE;
    WDTCN = 0xAD;             // выключение watchdog timer
    EA=1;
    OSCICN = 0x83;            // Настройка f=24.5 MHz
    PORT_sett ();
    preobr();
    while(1)
    {work ();}
}
//настройка портов
void PORT_sett (void)
{
    SFRPAGE = CONFIG_PAGE;
    XBR2 = 0x40;              //включаю матрицу ввода - вывода
    P3MDOUT=0xFF;
    P4MDOUT=0xFF;
    P5MDOUT=0xFF;
    P7MDOUT=0xFF;             //настройка портов P3-P5, P7 на вывод
}
void work (void)
{
    a=1;                      //запрет работы дешифратора
    b=0;                      //запрет работы регистра DD12
    P5=f;                     //тушение светодиодов
    P7=0x04;                  //включение регистра выбора индикатора
    b=1;                      //Разрешение работы регистра DD12
    a=0;                      //Разрешение работы дешифратора
}
void preobr (void)           // преобразование переменной d в
{                             //количество горящих светодиодов
    int i,c=0xFE;
    i=d;
    while (i>0)
    {f=f&c;                   // логическое И

```

}



ПРЕДВАРИТЕЛЬНОЕ ЗАДАНИЕ

1. Нарисовать схему включения 8 полупроводниковых индикаторов и рассчитать элементы схемы для индикаторов типа L-934SR в статическом режиме работы.
2. Нарисовать схему включения 8 полупроводниковых индикаторов и рассчитать элементы схемы для индикаторов типа L-934SR в динамическом режиме работы.
3. Представить диаграмму напряжений при работе индикаторов в динамическом и статическом импульсном режимах.

ПРОГРАММА РАБОТЫ

1. Изучить принципиальную схему лабораторного макета.
2. Подключить макетную плату к персональному компьютеру через разъем для подключения внешних устройств.
3. Составить алгоритм программы для вывода информации на индикаторы в статическом непрерывном режиме.

4. Написать программу вывода информации на индикаторы в статическом импульсном режиме. Вид выводимой информации задается преподавателем.

5. Составить алгоритм программы для вывода информации на индикаторы в динамическом режиме.

6. Написать программу вывода информации на индикаторы в динамическом режиме.

7. Снять осциллограммы напряжений для п.4 и п.6.

8. Определить критическую частоту мельканий индикаторов.

9. Определить пропускную способность оператора.

10. Определить пороговую чувствительность.

11. Определить разрешающую способность.

12. Измерить кажущуюся яркость индикаторов $L_{\text{вкж}}$, сравнить с расчетной.

13. Снять зависимость критической частоты мельканий от яркости.

СОДЕРЖАНИЕ ОТЧЕТА

1. Отчет должен содержать принципиальную схему, алгоритм и программу управления индикаторами.

2. Расчетные и экспериментальные значения напряжений и токов индикаторов в статическом режиме работы.

3. Диаграммы напряжений индикаторов в динамическом режиме работы. Сравнить экспериментальные и расчетные значения напряжений и токов, сделать выводы.

4. Расчет критической частоты мельканий.

5. Расчет пороговой чувствительности.

6. Расчет разрешающей способности.

7. Расчет пропускной способности оператора.

8. Построить зависимость критической частоты мельканий от яркости индикаторов.

9. Ответы на контрольные вопросы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Чем определяется цвет свечения п/п индикаторов?
2. Назовите основные характеристики п/п индикаторов.
3. Как определяется кажущаяся сила света при импульсном режиме работы индикатора?
4. Приведите классификацию дискретных индикаторов.

5. Чем определяются предельно допустимые значения прямого тока и напряжения?
6. Что такое пропускная способность оператора?
7. Чем определяется минимальная частота кадра?
8. Что такое период обновления информации?
9. Какими методами можно регулировать яркость свечения индикаторов?
10. Дайте определение: что такое дискретный индикатор.
11. Как определяется минимальный размер элемента отображения?
12. Чем отличается динамический режим работы дискретного индикатора от статического импульсного?
13. Как определяется скважность?
14. Перечислите преимущества однокоординатного метода адресации.

ЛАБОРАТОРНАЯ РАБОТА №2

ИЗУЧЕНИЕ ПРИНЦИПОВ ФОРМИРОВАНИЯ ИЗОБРАЖЕНИЯ НА ПОЛУПРОВОДНИКОВЫХ МНОГОРАЗРЯДНЫХ СЕМИСЕГМЕНТНЫХ ИНДИКАТОРАХ

Цель работы: изучение принципов формирования изображения на семисегментных индикаторах при двух-координатном методе управления и поразрядном способе индикации.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Дискретный индикатор это прибор, информационное поле которого состоит из отдельных фиксированных в пространстве элементов отображения (ЭО), а изображение создается одним ЭО или их совокупностью. Наиболее широко используемой группой дискретных индикаторов являются буквенно-цифровые приборы, которые позволяют отобразить арабские цифры от 0 до 9 и некоторые буквы русского алфавита. При отображении ограниченного количества знаков они позволяют построить более простую, экономичную, имеющую лучшие массогабаритные показатели и потребляющую меньшую мощность аппаратуру, чем на основе ЭЛТ. Для преобразования двоичной информации в код семисегментного индикатора используются либо дешифраторы, либо знакогенераторы, которые могут быть достаточно просто реализованы в микропроцессорных системах. Полупроводниковые индикаторы наиболее часто применяются для отображения знаковой информации при относительно небольших размерах символа и ограниченном числе знакомест. Эти приборы характеризуются высокой надежностью, низким уровнем управляющих напряжений и большой скоростью переключения.

Условное графическое изображение полупроводникового семисегментного индикатора с общим анодом приведено на рис.2.1.а, а с общим катодом - на рис.2.1.б. На рисунке 2.2 приведено расположение сегментов на информационном поле индикатора.

Для управления семисегментными индикаторами применяют в основном два метода адресации: однокоординатный и двухкоординатный.

Основным недостатком однокоординатного метода адресации дискретных индикаторов является большое число каналов управления:

$$N_y = N_{\text{эо}} \quad (1)$$

и выводов индикатора:

$$N_{\text{с}} = N_{\text{эо}} + 1 \quad (2)$$

Например, для семисегментного цифрового индикатора с десятичной точкой число выводов на один разряд равно $8+1=9$, а на три разряда уже 25. Из-за трудности создания индикаторов и схем управления с большим числом выводов и в особенности их соединения между собой применение методов однокоординатной адресации ограничивается цифровыми СОИ на три - четыре знакоместа. Поэтому при большом числе индикаторов широко применяют двухкоординатный метод адресации рис. 2.2.

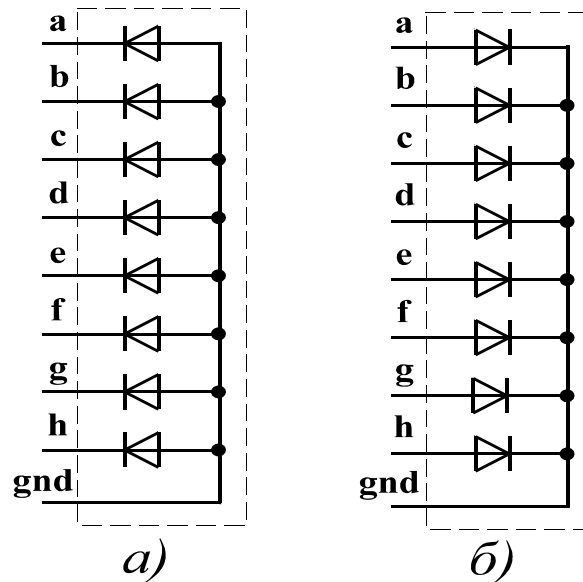


Рис.2.1. Условное графическое обозначение полупроводниковых индикаторов, а - с общим анодом, б - с общим катодом

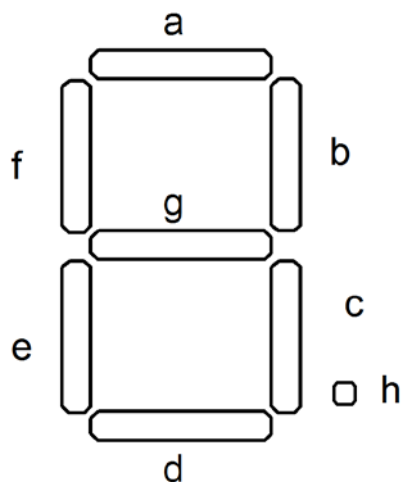


Рис.2.2. Расположение сегментов на информационном поле индикатора

Недостатком двухкоординатного метода адресации является то, что при переходе от одного разряда к другому происходит одновременное засвечивание сегментов в обоих разрядах, что ухудшает качество изображения. Для устранения такого недостатка применяют предварительное гашение сегментов при переходе с одного индикатора на другой (рис.2.3).

При двухкоординатном методе управления индикаторами возможно использование двух способов индикации:

- поразрядного,
- фазоимпульсного.

В данной лабораторной работе используется схема для поразрядного способа индикации. В этом случае кадр разбивается на столько тактов, сколько знакомест содержит блок индикации. В первом такте на сегменты индикаторов подается информация первого индикатора, во втором такте – соответственно информация второго индикатора и т.д.. При переходе с одного индикатора на другой производится гашение сегментов. Для обеспечения немелькающего изображения необходимо чтобы :

$$f_{\text{к}} \geq f_{\text{кчм}} \quad (4)$$

где : $f_{\text{кчм}}$ - критическая частота мелькания,
 $f_{\text{к}}$ - частота кадра.

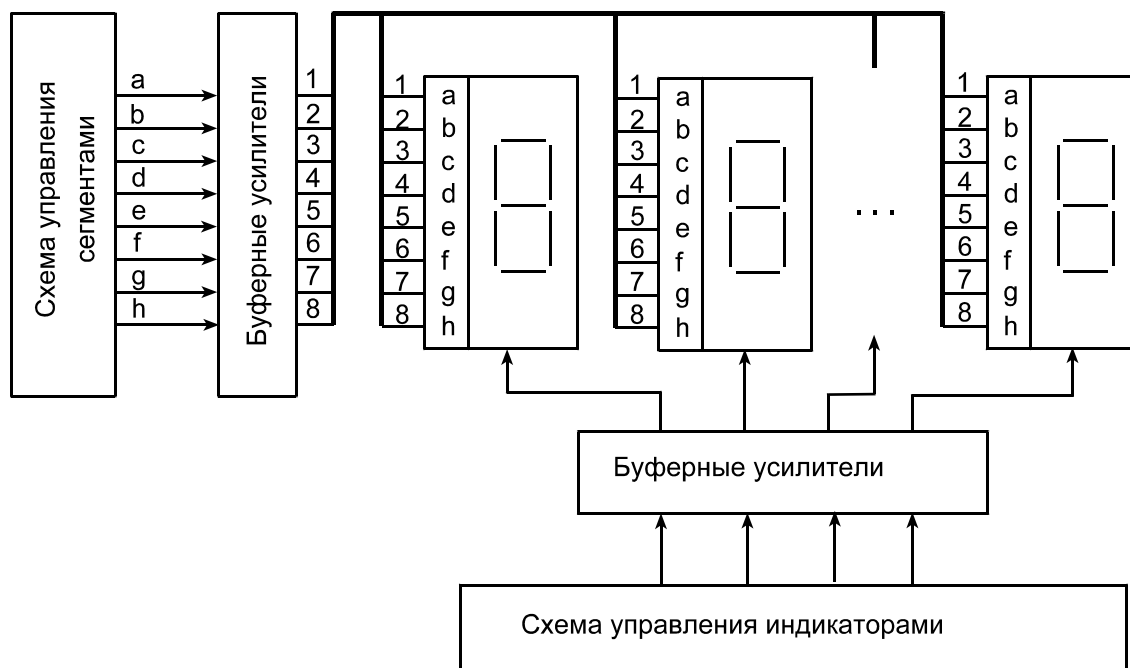


Рис.2.2. Подключение знаковых индикаторов при двухкоординатном методе адресации.

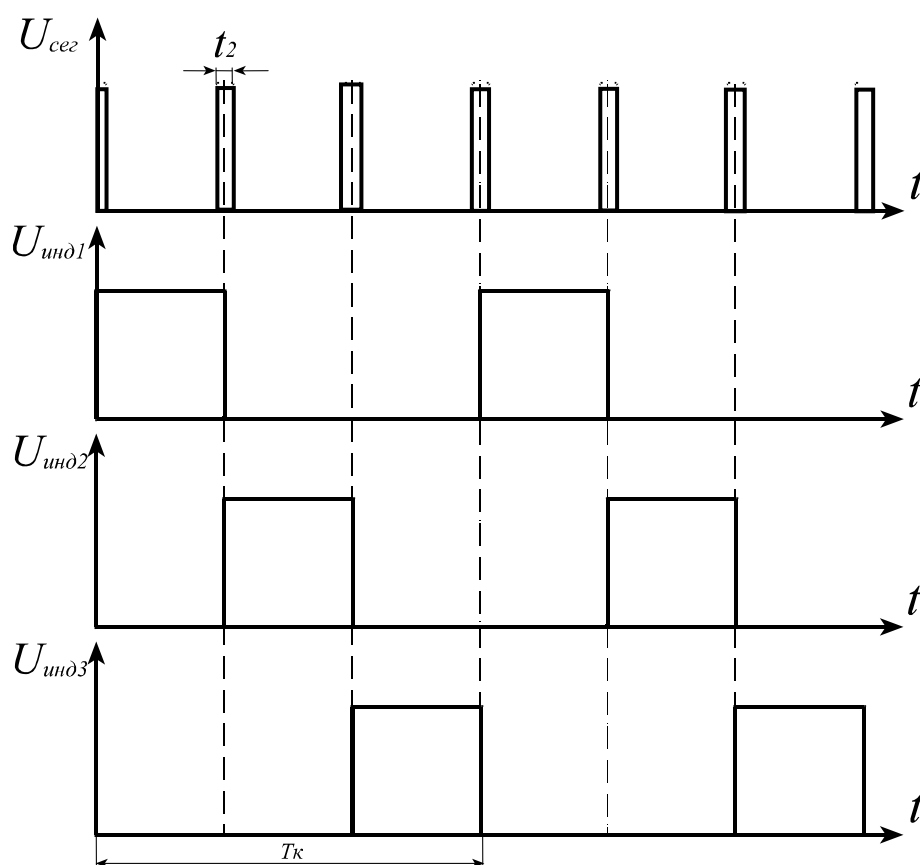


Рис. 2.3 Диаграммы работы трех семисегментных индикаторов в динамическом режиме работы.

ОПИСАНИЕ ЛАБОРАТОРНОЙ УСТАНОВКИ

В лабораторной работе используется стенд на базе микроконтроллера, обеспечивающего управление полупроводниковыми семисегментными индикаторами через буферные регистры. В качестве индикаторов используются сдвоенные светодиодные индикаторы типа DA56-11RWA с общим анодом и DC56-11RWA с общим катодом. Их основные параметры приведены в таблице 2.1, а вольт-амперная и кандел-амперная характеристики для индикаторов разного цвета свечения - на рисунке 2.4а и 2.4б соответственно. На рисунке 2.5 показан внешний вид индикатора DA56-11.

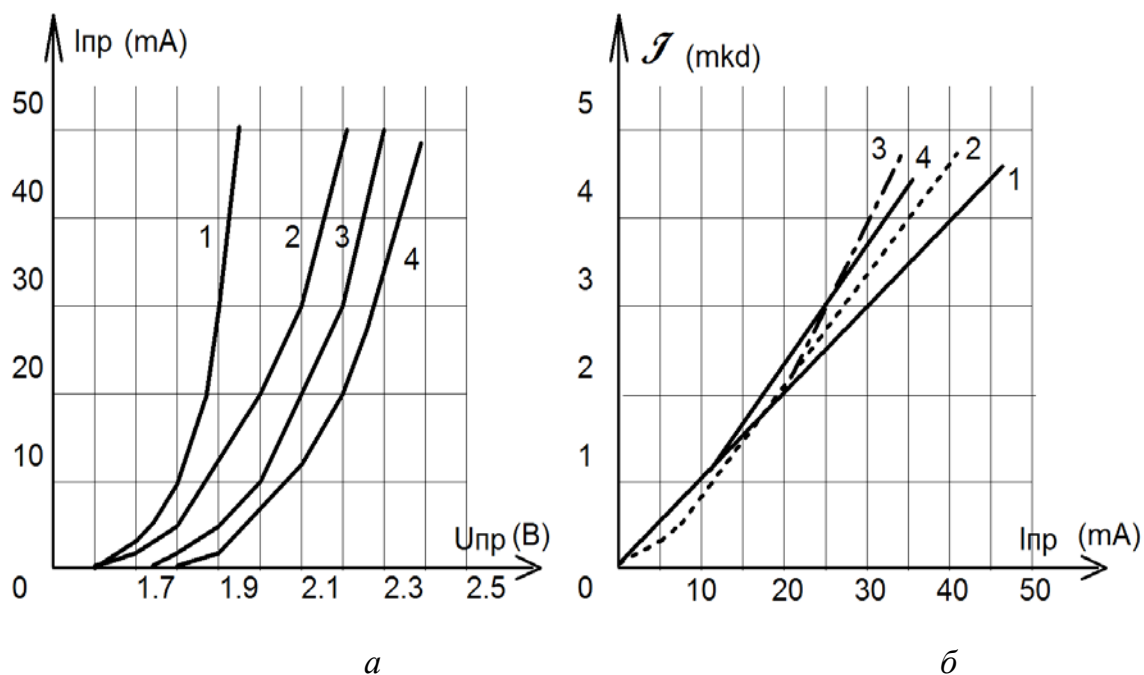


Рис.2.4. Характеристики полупроводниковых индикаторов DA56-11, а – вольт-амперная, б – кандел-амперная, 1 – суперяркий красный, 2 – красный, 3 – зеленый, 4 – желтый

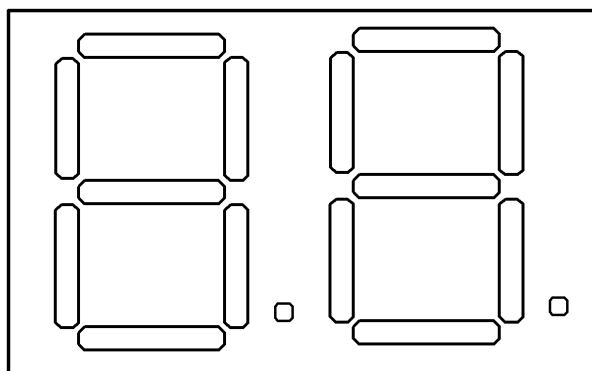


Рис.2.5. Внешний вид светодиодного индикатора DA56-11

В лабораторной установке используются семисегментные индикаторы как с общим катодом DC56-11RWA рис 2.6 так и с общим анодом DA56-11RWA рис.2.7. Сегменты индикаторов через буферный регистр DD13 и регистр памяти DD16 подключены к порту P5 микроконтроллера C8051F060. Общий электрод индикатора (общий катод для индикатора DC56-11RWA и общий анод для индикатора DA56-11RWA) подключен через регистр памяти DD17, буферный регистр DD13 к порту P5 микроконтроллера C8051F060. **Выбор индикатора осуществляется путем записи данных в регистр DD17.** Загрузка данных в регистр DD17 осуществляется установкой кода 00000101b на выходах порта AD и подачей импульса ALE. При появлении этого импульса данные с порта AD записываются в регистр DD12 и поступают на вход дешифратора DD23, при подаче сигнала IWR с порта P3.5 на пятом выходе дешифратора появляется активный уровень, который поступает на тактовый вход регистра DD17 и записывает в него выходные данные порта P5. **Выбор сегментов индикатора осуществляется путем записи данных в регистр DD16.** Загрузка данных в регистр DD16 осуществляется установкой кода 00000000b на выходах порта AD и подачей импульса ALE. При появлении этого импульса данные с порта AD записываются в регистр DD12 и поступают на вход дешифратора DD23, при подаче сигнала IWR с порта P3.5 на нулевом выходе дешифратора появляется активный уровень, который поступает на тактовый вход регистра DD16 и записывает в него выходные данные порта P5.

При использовании индикаторов с общим катодом (рис.2.6) активным уровнем для выбора индикатора является логическая единица. Для засвечивания соответствующего индикатора необходимо записать уровень логической единицы на соответствующий выход регистра DD17, которая поступает в базу соответствующего транзистора структуры **n-p-n** и открывает его. Динамический режим

работы индикаторов обеспечивается бегущей единицей на выходах регистра DD17. Кодировка разрядов индикатора приведена в таблице 2.2.

Основные параметры индикаторов DX56-11.

Таблица 2.1.

Параметр	DX56-11EWA	DX56-11GWA	DX56-11YWA	DX56-11SRWA
Длина волны (нм)	625	568	588	640
Постоянный прямой ток (mA)	30	25	30	30
Импульсный прямой ток (mA)	160	140	140	155
Прямое напряжение (В)	2,0-2,5	2,2-2,5	2,1-2,5	1,85-2,5
Обратный ток (μA)	10	10	10	10
Обратное напряжение (В)	5	5	5	5
Сила света (μkd)	1900-8000	3000-10500	1900-4700	8000-24000
Рассеиваемая мощность мВт	105	105	105	100
Диапазон рабочих температур	-40 - +85	-40 - +85	-40 - +85	-40 - +85

Активному уровню каждого сегмента соответствует уровень логической единицы на соответствующем выходе регистра DD16. Кодировка сегментов индикатора приведена в таблице 2.3. Для выбора нулевого разряда необходимо в регистр DD17 записать число 10h, при выборе третьего разряда - в регистр нужно записать число 80h.

Для одновременного выбора нескольких сегментов необходимо произвести операцию дизъюнкции с кодами выбранных сегментов. Например, для вывода цифры 4 нужны сегменты b, c, f и g. Выполняем операцию дизъюнкции с кодами этих сегментов

$$02h + 04h + 20h + 40h = 66h$$

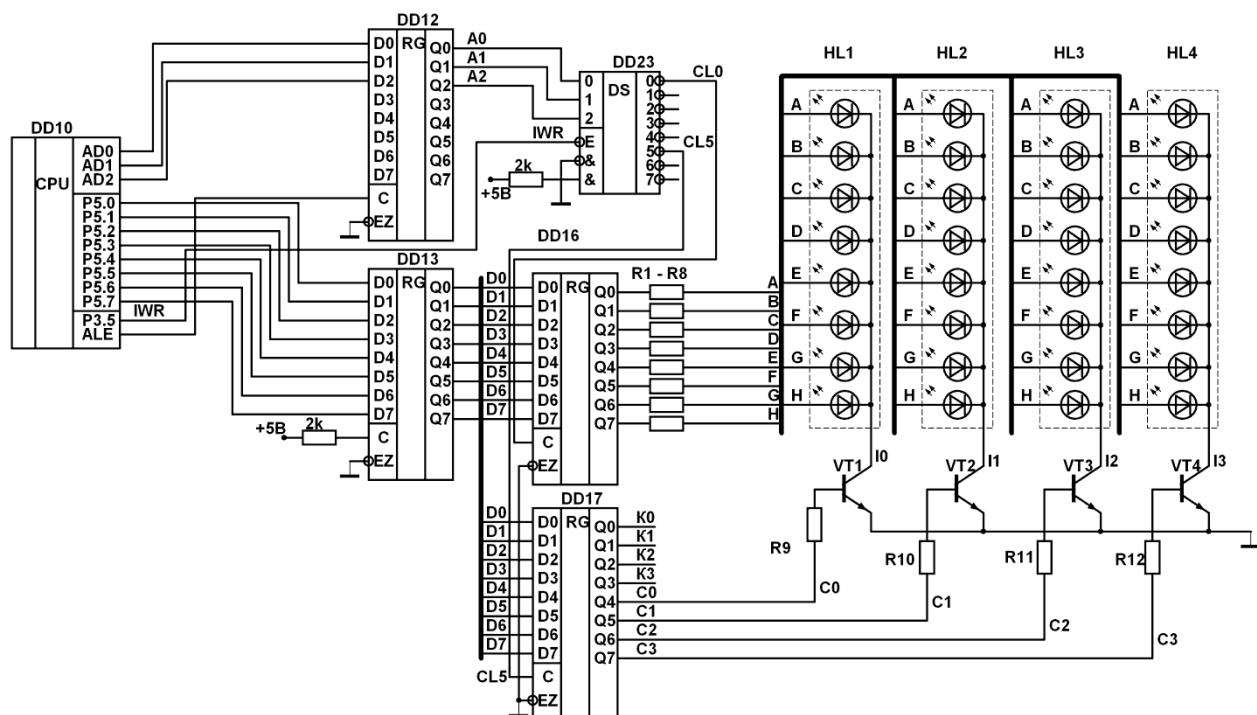


Рис.2.6.Схема управления 4-разрядным семисегментным полупроводниковым индикатором с общим катодом

Кодировка разрядов индикатора.

Таблица 2.2

Разряд	0	1	2	3
Код	00010000b	00100000b	01000000b	10000000b

Кодировка сегментов индикатора.

Таблица 2.3

Сегмент	A	B	C	D	E	F	G	H
Код	01H	02H	04H	08H	10H	20H	40H	80H

Полученный код записываем в регистр DD16. В таблице 2.4. представлен результат преобразования цифр в коды семисегментного индикатора с общим катодом.

При использовании индикаторов с общим анодом (рис.2.7) активным уровнем для выбора индикатора является логический ноль. Для засвечивания соответствующего индикатора необходимо записать уровень логического нуля на соответствующий выход регистра DD17, который поступает в базу соответствующего транзистора структуры **p-**

n-p и открывает его. Динамический режим работы индикаторов обеспечивается бегущим нулем на выходах регистра DD17. Кодировка разрядов индикатора приведена в таблице 2.5.

Кодировка цифр в коды семисегментного индикатора с общим катодом

Таблица 2.4

цифра	1	2	3	4	5	6	7	8	9	0
Код	06h	5Bh	4Fh	66h	6Dh	7Dh	07h	7Fh	6Fh	3Fh

Активному уровню каждого сегмента соответствует уровень логического нуля на соответствующем выходе регистра DD16. Запись данных в регистр DD16 производится по той же методике, что и для индикаторов с общим катодом. Кодировка сегментов индикатора приведена в таблице 2.6.

Кодировка разрядов индикатора.

Таблица 2.5

Разряд	0	1	2	3
Код	11101111b	11011111b	10111111b	01111111b

Для выбора нулевого разряда необходимо в регистр DD17 записать число 0EFH, при выборе третьего разряда - в регистр нужно записать число 07Fh.

Кодировка сегментов индикатора.

Таблица 2.6

Сегмент	A	B	C	D	E	F	G	H
Код	0FEH	0FDH	0FBH	0F7H	0EFH	0DFH	0BFH	7FH

Для одновременного выбора нескольких сегментов необходимо произвести операцию конъюнкции с кодами выбранных сегментов. Например, для вывода цифры 4 нужны сегменты b, c, f и g. Выполняем операцию конъюнкции с кодами этих сегментов

$$0FDh \wedge 0FBh \wedge 0DFh \wedge 0BFh = 99h$$

Полученный код записываем в регистр DD16. В таблице 2.7. представлен результат преобразования цифр в коды семисегментного индикатора с общим анодом.

Кодировка цифр в коды семисегментного индикатора с общим анодом

Таблица 2.7

цифра	1	2	3	4	5	6	7	8	9	0
Код	F9h	A4h	B0h	99h	92h	82h	F8h	10h	90h	C0h

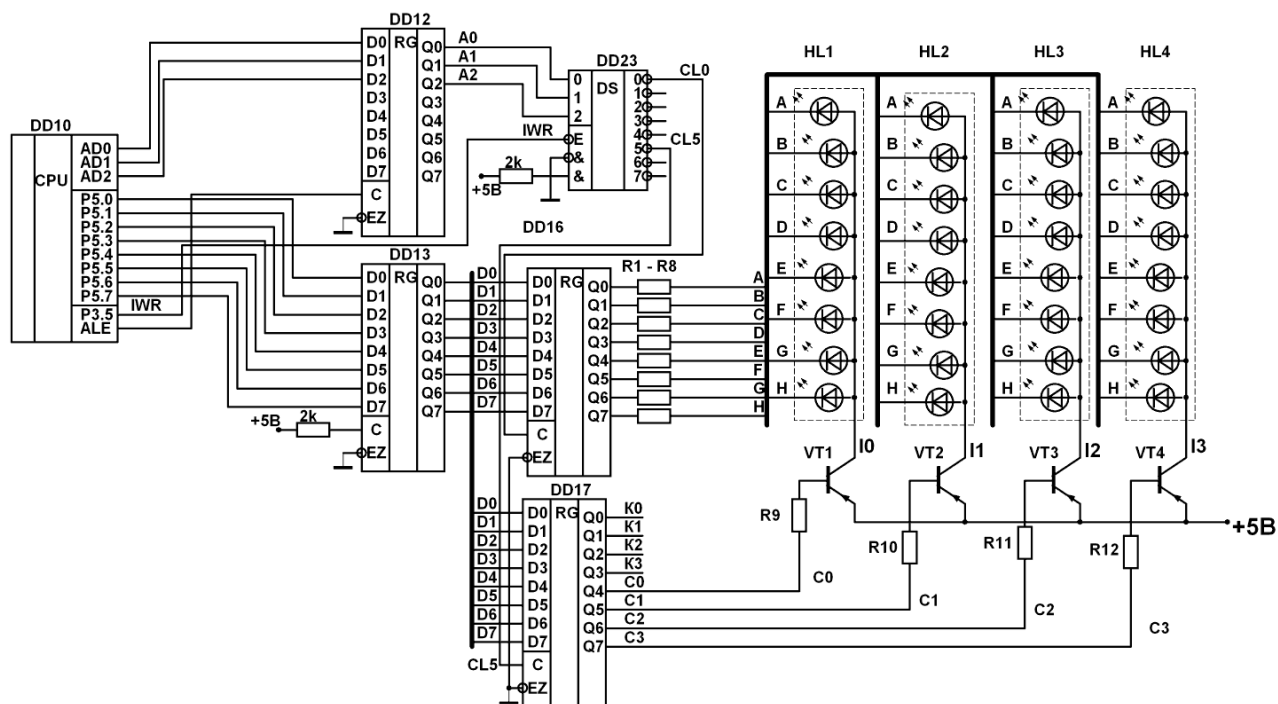


Рис.2.7.Схема управления 4-разрядным семисегментным полупроводниковым индикатором с общим анодом

В качестве примера ниже приведен алгоритм (рис.2.8) и программа для отображения числа 1986 на четырех индикаторах. Длительность задержки, равная времени выборки одного индикатора, выбирается из условия:

$$t_3 = t_B \leq \frac{T_K}{N} = \frac{1}{N * f_{кр}}$$

где:

T_K - период кадра, N – количество индикаторов, $f_{кр}$ - критическая частота мельканий.

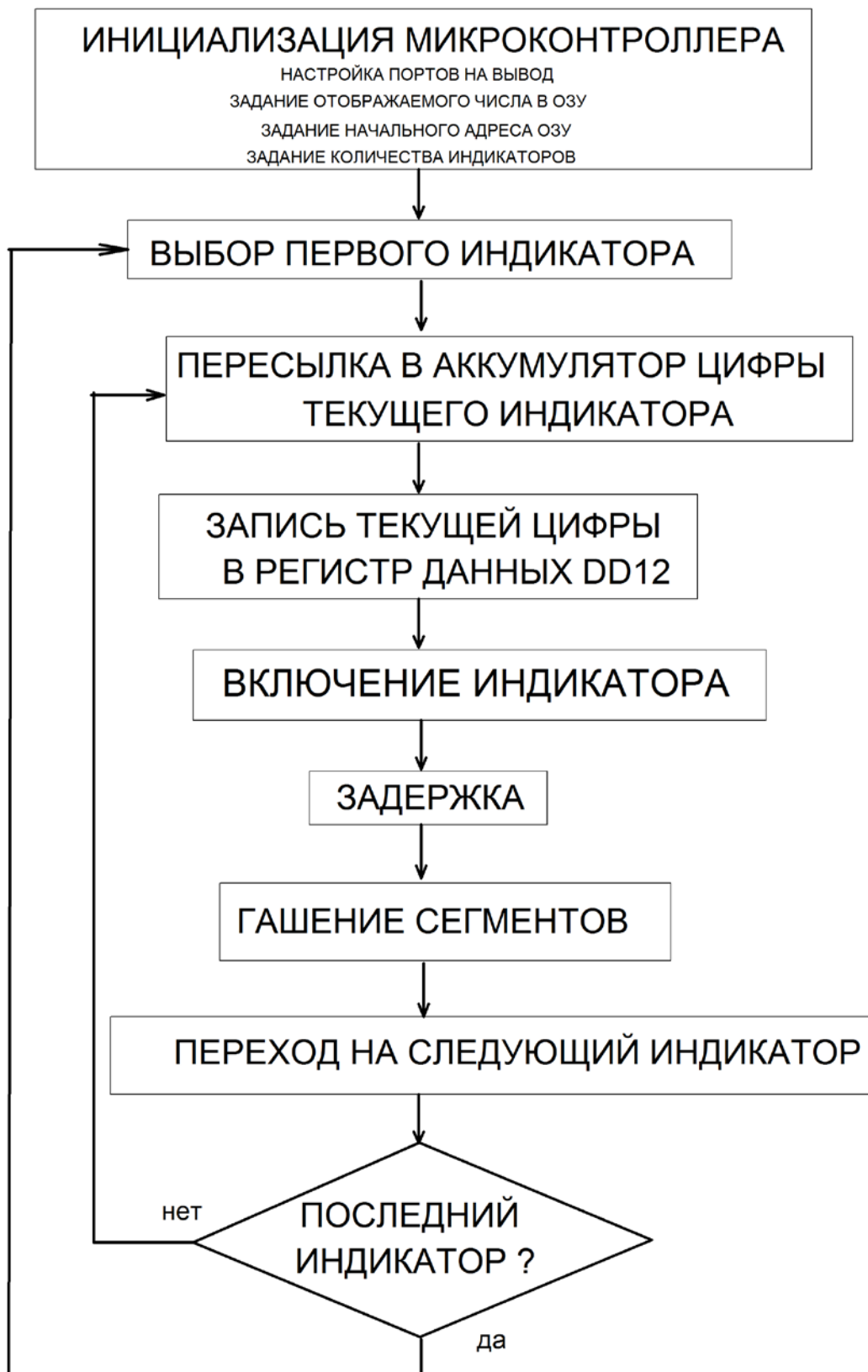


Рис.2.8. Алгоритм программы для отображения числа 1986 на четырех индикаторах

Пример программы на разных языка программирования, реализующий приведенный выше алгоритм, для отображения числа 1986 на четырех индикаторах приведены ниже.

- код программы на языке ассемблера:

```
$INCLUDE (C8051F060.inc)
    R8    EQU 0X08    ;ЗАДАНИЕ РЕГИСТРА
;-----
; настройка стека
;-----
STACK SEGMENT IDATA ;выбор сегмента стека
RSEG STACK
DS 80h    ;резервирование 128 байтов для стека
;-----
; настройка таймеров
;-----
    CSEG AT 0
    ljmp  Main

Temp_2    SEGMENT CODE ;
    RSEG Temp_2    ;выбор сегмента CODE
    USING 0    ;использование нулевого банка регистров

Main:
    mov  WDTCN, #0deh ;выключение watchdog timer
    mov  WDTCN, #0adh
    mov  SP, #STACK-1 ;инициализация указателя стека

    mov  SFRPAGE, #0FH    ;выбор страницы 0fH
    MOV  XBR2, #40H

    MOV  P3MDOUT, #0FFH ;
    MOV  P4MDOUT, #0FFH ;
    MOV  P5MDOUT, #0FFH ;
    MOV  P7MDOUT, #0FFH ;настройка портов на вывод
GOO:
;-----
;ВВОД НАЧАЛЬНЫХ ЗНАЧЕНИЙ
;-----
    MOV  R2, #11111001B    ;цифра 1
    MOV  R3, #10010000B    ; цифра 9
```

```

MOV R4,#10000000B    ;цифра 8
MOV R5,#10000010B    ; цифра 6
MOV R6,#11111111B    ;гашение сегментов индикаторов
MOV R7,#11101111B    ;выбор первого индикатора
MOV R8,#04H          ;количество индикаторов
MOV R0,#02           ;
MOV R1,#0X           ;задание начального адреса
;-----
;основная программа
;-----
GO:
    MOV A,@R0          ;пересылка в аккумулятор числа
    SETB P3.5          ;запрет работы дешифратора DD23
    CLR P4.5           ;запрет работы регистра DD12

    MOV P5,A           ;выбор символа
    MOV P7,#11111000b ;включение регистра выбора символа

    setb P4.5          ;разрешение работы регистра DD12
    clr P3.5           ; разрешение работы дешифратора DD23

    SETB P3.5          ; запрет работы дешифратора DD23
    CLR P4.5           ; запрет работы регистра DD12

    mov  P5,R7          ;выбор индикатора
    mov  P7,#11111101b ;включение регистра выбора индикатора

    setb P4.5          ; разрешение работы регистра DD12
    clr P3.5           ; разрешение работы дешифратора DD23
;-----
-
    MOV A,R7 ;
    SWAP     A  ;
    RL      A  ;
    SWAP     A  ;
    MOV R7,A   ;переход на следующий индикатор
;-----
    SETB P3.5    ; запрет работы дешифратора DD23
    CLR P4.5     ; запрет работы регистра DD12

    MOV P5,R6     ;гашение сегментов индикаторов

```

```

MOV P7,#11111000b    ;включение регистра выбора символа

setb P4.5            ; разрешение работы регистра DD12
clr P3.5             ; запрет работы регистра DD12

SETB P3.5           ; запрет работы дешифратора DD23
CLR P4.5            ; запрет работы регистра DD12

mov  P5,R7          ;выбор индикатора
mov  P7,#1111101b   ;включение регистра выбора индикатора

setb P4.5            ; разрешение работы регистра DD12
clr P3.5             ; запрет работы регистра DD12
;-----
;ЦИКЛ
;-----
    INC  R0          ;выбор следующего символа
    DJNZ R8,GO        ;проверка количества индикаторов
    JMP  GOO          ;повтор программы
    NOP
    END

```

- код программы на языке C++:

```

#include "C8051F060.h"
sbit a=P3^5;
sbit b=P4^5;
void PORT_sett (void);
void work (void);
void preobr (void);
void TIM_sett (void);
void delay (unsigned int del);
int v[]={0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x10, 0x90};
    // преобразуемые цифры
int d[]={2, 3, 4, 5};                // вводите сюда данные
int svet[]={0xDF, 0xBF, 0x7F, 0xEF}; //адреса индикаторов
int i, p;
//p - переменная в которой храниться преобразуемое значение
//i - счетчик сегментных индикаторов
void main (void)

```

```

{
    SFRPAGE = CONFIG_PAGE;
    EA=0;                                //
    WDTCN = 0xDE; // выключение watchdog timer
    WDTCN = 0xAD;
    EA=1;
    OSCICN = 0x83;                       // Настройка f=24.5 MHz
    PORT_sett ();
    while(1)
    {work ();}
}
//настройка портов
void PORT_sett (void)
{
    SFRPAGE = CONFIG_PAGE;
    XBR2 = 0x40;    //включаю матрицу ввода - вывода
    P3MDOUT=0xFF;
    P4MDOUT=0xFF;
    P5MDOUT=0xFF;
    P7MDOUT=0xFF;    //настройка портов P3-P5, P7 на вывод
}
//настройка таймера для формирования задержки
void TIM_sett (void)
{
    SFRPAGE = TIMER01_PAGE;
    TMOD |= 0x01;    //для формирования 1 мс
}
void work (void)
{
    i=0;
    while (i<4)
    {preobr ();}
}
//-----
//Цыфра горит
    a=1;    //запрет работы дешифратора
    b=0;    //запрет работы регистра DD12
    P5=p;    //горит цыфра
    P7=0xF8;    // выбор светодиодов

```

```

b=1;          //Разрешение работы регистра DD12
a=0;          //Разрешение работы дешифратора
a=1;          //запрет работы дешифратора
b=0;          //запрет работы регистра DD12
P5=svet[i];   //выбор индикатора
P7=0xFD;
b=1;          //Разрешение работы регистра DD12
a=0;          //Разрешение работы дешифратора
delay (250);  // формирование задержки
//цифра не горит!!
a=1;          //запрет работы дешифратора
b=0;          //запрет работы регистра DD12
P5=0xFF;      //тушение светодиодов
P7=0xF8;      // выбор светодиодов
b=1;          //Разрешение работы регистра DD12
a=0;          //Разрешение работы дешифратора
a=1;          //запрет работы дешифратора
b=0;          //запрет работы регистра DD12
P5=svet[i];   //выбор индикатора
P7=0xFD;
b=1;          //Разрешение работы регистра DD12
a=0;          //Разрешение работы дешифратора
//-----
i++;          //следующий разряд
}
void preobr (void)
// ввод: p преобразуемый символ, вывод: p- код на семисегментнике
{
    int t;
    t=0;
    while (t<10)
    {
        if (t==d[i]) // ищем что за цифра
        {
            p=v[t]; // загрузка кода цифры
            t++;
        }
    }

    //формирователь задержек

```



```

void delay (unsigned int del)
{
    SFRPAGE = TIMER01_PAGE;
    //для мс
    TH0 = 0xFE;
    TL0 = 0x0B;
    TR0=1;
    while (del!=0)
    {
        while (!TF0){}
        TH0 = 0xFE;
        TL0 = 0x0B;
        del--;
        TF0=0;    }
    TR0=0;
    SFRPAGE = CONFIG_PAGE;
}

```

При обновлении информации необходимо чтобы период обновления информации $T_{обн}$ был больше пропускной способности оператора.

ПРОГРАММА РАБОТЫ

1. Подключить лабораторный стенд к персональному компьютеру через разъемы для внешних устройств.
2. Записать таблицу преобразования шестнадцатичных чисел в коды семисегментного индикатора.
3. Составить алгоритм вывода информации на индикаторы.
4. Составить программу вывода информации на индикаторы. Для каждого индикатора выделить одну ячейку ОЗУ для хранения данных.
5. Записать в выделенные ячейки ОЗУ следующие данные:

для первого индикатора	01D
для второго индикатора	02D
для третьего индикатора	03D
для четвертого индикатора	04D
6. Преобразовать эту информацию и вывести на индикаторы.
7. Снять осциллограммы напряжений на сегментах индикаторов.
8. Снять осциллограммы напряжений на общем электроде индикатора.
9. Измерить время выборки одного индикатора.

10. Измерить время гашения сегментов.
11. Определить критическую частоту мелькания изображения.
12. Написать программу смены информации в ОЗУ.
13. Вывести на индикаторы изменяющуюся информацию (бегущую строку, секундомер).
14. Определить пропускную способность оператора

СОДЕРЖАНИЕ ОТЧЕТА

Отчет должен содержать:

1. Структурную схему блока индикации для двухкоординатного метода управления.
2. Принципиальную схему блока управления четырехразрядным полупроводниковым индикатором с общим анодом для двухкоординатного метода управления и поразрядного метода индикации.
3. Принципиальную схему блока управления четырехразрядным полупроводниковым индикатором с общим анодом для двухкоординатного метода управления и фазоимпульсного метода индикации.
4. Расчет элементов схемы управления индикаторами для двух методов индикации при $I_{сегм.} = 10 \text{ mA.}$, $U_{пр} = 2 \text{ В.}$
5. Алгоритм и программу вывода информации.
6. Осциллограммы напряжений на разрядах и сегментах индикаторов.
7. Расчет критической частоты мельканий.
8. Расчет пропускной способности оператора.
9. Выводы.
10. Ответы на контрольные вопросы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение динамическому режиму работы индикаторов.
2. Как определяется кажущаяся яркость при динамическом режиме работы индикатора?
3. Что такое период кадра?
4. Чему равна скважность при динамическом режиме работы индикаторов?

5. Нарисуйте структурную схему управления индикаторами при двухкоординатном способе адресации.

6. Дайте определение времени включения и времени выключения индикатора.

7. Какой метод индикации применен в лабораторной работе, какие еще методы существуют?

8. Нарисуйте принципиальную схему управления полупроводниковыми индикаторами для двухкоординатного метода управления.

9. Перечислите преимущества и недостатки двухкоординатного метода адресации по сравнению с однокоординатным.

ЛАБОРАТОРНАЯ РАБОТА №3

ИЗУЧЕНИЕ ПРИНЦИПОВ ФОРМИРОВАНИЯ ИЗОБРАЖЕНИЯ НА ЖИДКОКРИСТАЛЛИЧЕСКИХ МНОГОРАЗРЯДНЫХ СЕМИСЕГМЕНТНЫХ ИНДИКАТОРАХ

Цель работы: изучение принципов формирования изображения на жидкокристаллических семисегментных индикаторах при двухкоординатном методе управления и поразрядном способе индикации.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В противоположность активным приборам жидкокристаллические (ЖК) индикаторы не генерируют свет, а только управляют его прохождением, что обуславливает чрезвычайно малую потребляемую ими мощность. Преимуществом ЖК-индикаторов является также малое управляющее напряжение, позволяющее непосредственно согласовывать их с цифровыми интегральными схемами. Так как ЖК-индикаторы относятся к классу пассивных, то основным их оптическим параметром является не яркость, а контрастность (для просветных индикаторов вместо контрастности часто пользуются коэффициентом пропускания, который определяют как отношение интенсивностей выходящего света к падающему).

Вольт-контрастная характеристика жидкокристаллического индикатора зависит не от амплитудного, а от действующего значения приложенного напряжения.

Типичные вольт-контрастные характеристики твист-индикатора для углов наблюдения 0 и 45° показаны на рис. 3.1. Для параметрического задания кривой можно взять напряжения, соответствующие 5, 10, 50 и 90% контраста, обозначая их $U_5, U_{10}, U_{50}, U_{90}$.

Относительная обратная крутизна электрооптической характеристики

$$\gamma_{90-10} = \frac{(U_{90} - U_{10})}{U_{10}}; \gamma_{50-10} = \frac{(U_{50} - U_{10})}{U_{10}};$$
$$\gamma_{50-5} = \frac{(U_{50} - U_5)}{U_5}$$

Динамические параметры ЖК-индикаторов определяются временами реакции τ_{pk} и релаксации τ_{pl} . Кривая изменения контраста при наложении импульса напряжения имеет следующие участки:

- задержку включения,
- время нарастания (в сумме они равны времени реакции),
- задержку выключения (обычно очень малую),
- время спада (две последние составляющие в сумме дают время релаксации).

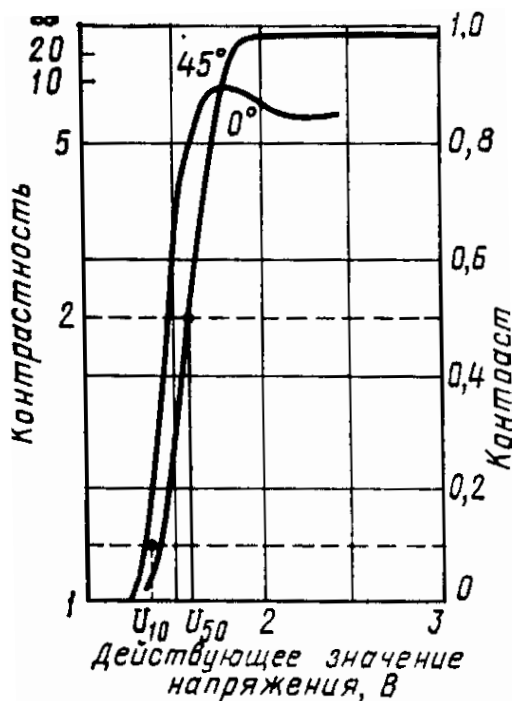


Рис. 3.1. Вольт-контрастные характеристики твист-индикаторов

Условное графическое изображение жидкокристаллического семисегментного индикатора приведено на рис.3.2.

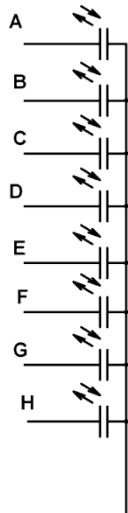
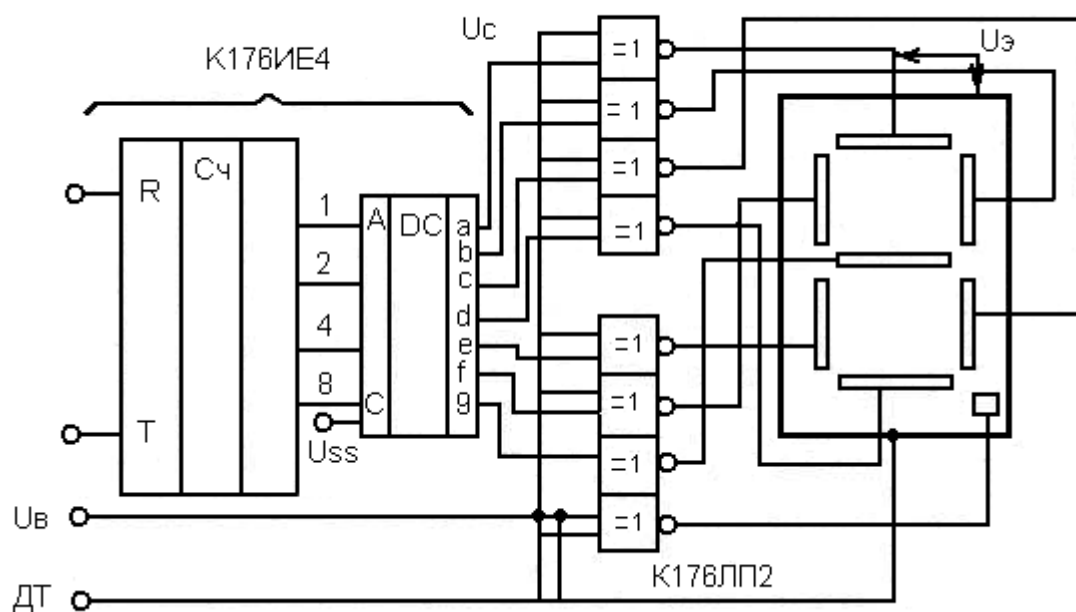
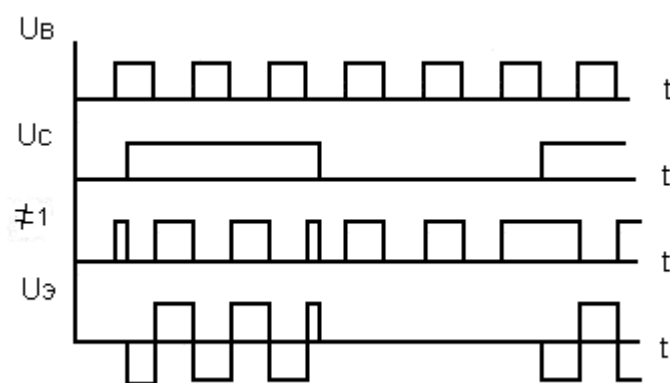


Рис.3.2. Условное графическое обозначение семисегментного жидкокристаллического индикатора

Наиболее распространен фазовый способ управления, основанный на применении логической схемы «исключающее ИЛИ» ($f = A\bar{B} \vee \bar{A}B$, где f — выходная функция, а A и B — входные переменные). Такая схема показана на рис. 3.3,а, а диаграммы сигналов для включенного и выключенного состояний жидкокристаллического индикатора, — на рис. 3.3,б. Преимуществом схемы является почти двукратное повышение напряжения между электродами индикатора $U_э$ по сравнению с напряжением питания ИМС E_k .



а)



б)

Рис. 3.3. Фазоимпульсный способ управления знаковым ЖК-индикатором: а — схема включения; б — диаграммы напряжений

Импульсное напряжение U_B , используемое в схеме рис. 3.3,б, должно иметь частоту, превышающую частоту мельканий, и в то же время быть значительно ниже частоты отсечки, при которой пороговое напряжение резко возрастает. Одновременное воздействие на индикатор слишком высокого напряжения и частоты приводит к перевозбуждению, что вызывает увеличение времени релаксации и возникновение эффекта креста (из-за внутренних падений напряжения на электродах индикатора). В результате при выборе рабочей частоты должно выполняться условие

$$f * U < K_{np},$$

где $K_{np} = 1000 \div 3000$ (предельное значение для большинства типов индикаторов).

Для двухкоординатного метода адресации ЖК индикаторов удобно применять схемы управления с Z-состоянием рис.3.4. Строки индикатора соединяются вместе, а столбцы подключены к логическим элементам с Z-состоянием. Управление Z-состоянием логических элементов обеспечивает дешифратор позиционного кода DD14, который разрешает поочередное включение логических элементов.

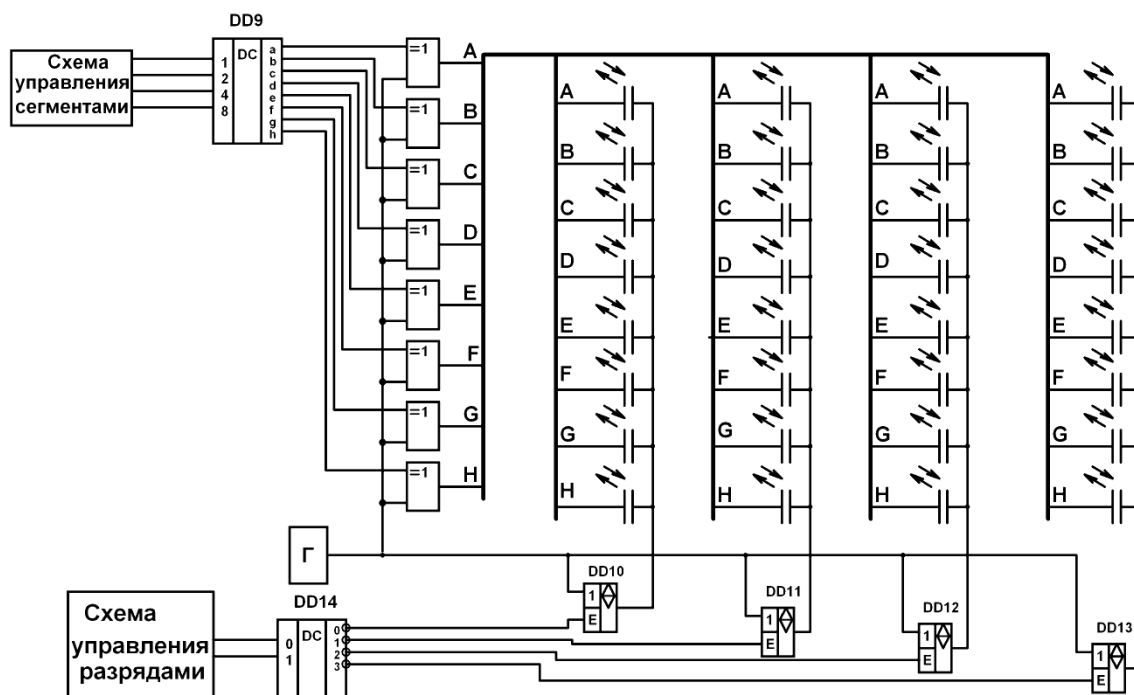


Рис.3.4. Подключение жидкокристаллических элементов отображения при двухкоординатном методе адресации.

ОПИСАНИЕ ЛАБОРАТОРНОЙ УСТАНОВКИ

В лабораторной работе используется стенд на базе микропроцессорного комплекта, обеспечивающего управление жидкокристаллическими семисегментными индикаторами. В лабораторной работе используются 2 жидкокристаллических индикатора типа DE-112RS рис 3.4. Их основные параметры приведены в таблице 3.1.

Сегменты индикаторов через буферный усилитель DD14 подключены к порту P2 микроконтроллера C8051F060. Общий электрод индикаторов подключен к 7 разряду порта P2 рис. 3.5. **Выбор индикатора осуществляется путем записи данных в порт P3.0.** Логический ноль подключает первый и второй разряды, логическая единица подключает третий и четвертый разряды индикаторов. **Выбор сегментов индикатора осуществляется путем записи данных в буферное устройство DD14.** Для настройки буферного устройства на передачу данных от порта P2 к сегментам необходимо установить на выходе порта P3.1 уровень логической единицы. Для разрешения работы буферного устройства необходимо установить уровень логического нуля на выходе порта P3.3.

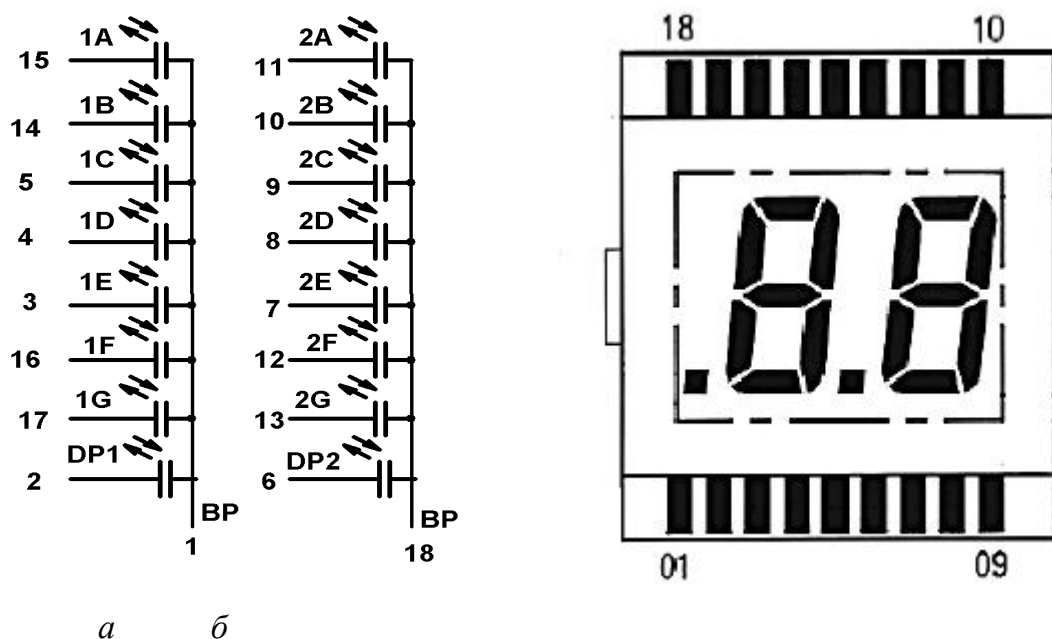


Рис. 3.4. Жидкокристаллический индикатор DE 112RS, а - условное графическое обозначение, б - сборочный чертеж

Динамический режим работы индикаторов обеспечивается инвертированием данных на выходе P3.0. Кодировка разрядов индикатора приведена в таблице 3.2.

Активному уровню каждого сегмента соответствует инверсное значение данных на выходе порта P2.7. Кодировка сегментов индикатора приведена в таблице 3.3.

Основные параметры ЖК индикатора DE 112RS.

Таблица 3.1.

Параметр\значение	min	typ	max	Единица измерения
Пороговое напряжение		3		В
Частота управляющего напряжения	30	32	100	Гц
Плотность тока		1,0	2,0	$\mu A/ см^2$
Время реакции ($t_{on} + t_{off}$)		440		мс
Диапазон рабочих температур	-10		+60	$^{\circ}C$
Долговечность		100 000		час

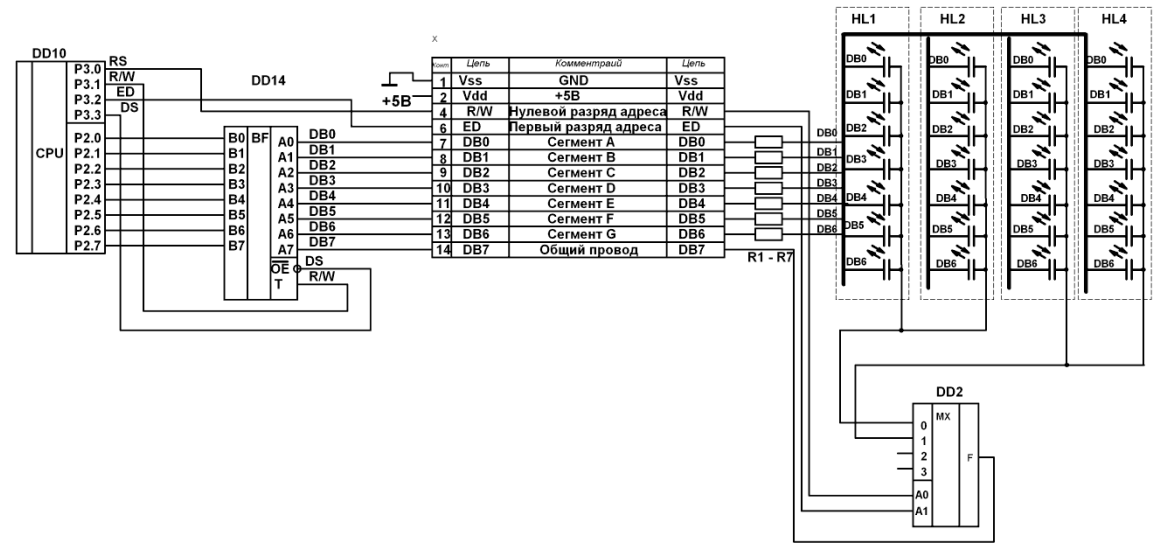


Рис.3.5.Схема управления 4-разрядным семисегментным жидкокристаллическим индикатором

Кодировка разрядов индикатора.

Таблица 3.1

Разряд	0	1	2	3
Код	00000000b	00000000b	00000001b	00000001b

Для выбора нулевого разряда необходимо на выход порта P3.0 записать число 00H, при выборе второго разряда - 01h.

Кодировка сегментов индикатора.

Таблица 3.2

Сегмент	A	B	C	D	E	F	G
Код	01H	02H	04H	08H	10H	20H	40H

Для одновременного выбора нескольких сегментов необходимо произвести операцию дизъюнкции с кодами выбранных сегментов. Например, для вывода цифры 4 нужны сегменты b, c, f и g. Выполняем операцию дизъюнкции с кодами этих сегментов:

$$02h + 04h + 20h + 40h = 66h$$

Полученный код записываем в регистр DD14.

В качестве примера ниже приведен алгоритм для управления жидкокристаллическим индикатором рис.3.6.

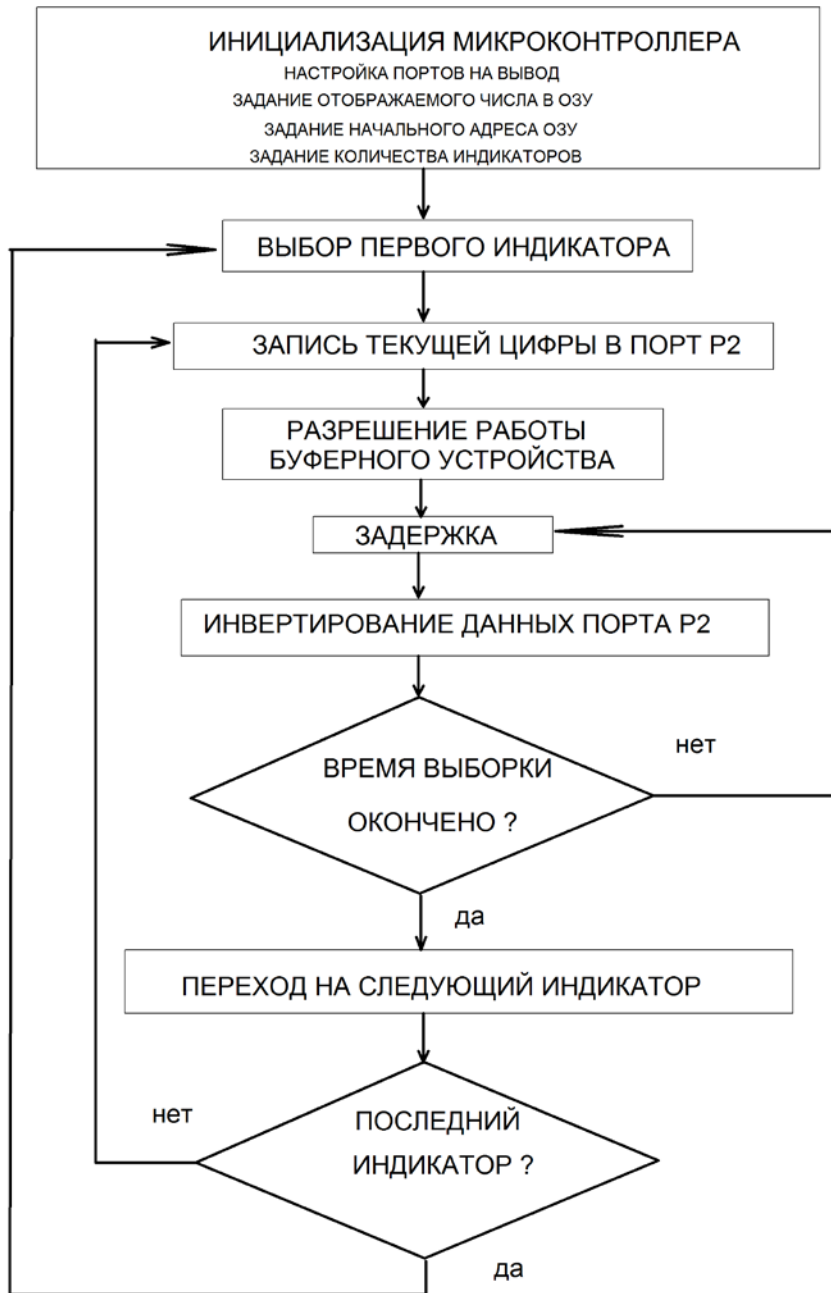


Рис.3.6. Алгоритм управления жидкокристаллическим индикатором

Программная реализация приведенного на рисунке 3.6 алгоритма приведена ниже. Программа позволяет вывести цифру 5 в первый разряд индикатора.

```

;-----
; Lab_Test.ASM программа учебно-лабораторног учебного стенда.
; Команда $8F - программный сброс контроллера. Ответ: 70$,'O','K','S','b','t',CRC8
; Команда $80 - Запрос версии ПО в контроллере. Ответ 7F$,'1','4','N','1','0',CRC8
; Команда $81,D5,D7,D8,P0.7,P0.6,x,CRC8 - вывод данных в соответствующие регистры.
; Команда /$82,D5,D7,D8,P0.7,P0.6,x,CRC8 - Запрос значений в регистрах D5,D7,D8.
; Команда $83,x,x,x,x,N,CRC8 Запрос N измерения данных с ADC0 и ADC1.

```

```

;Команда вывести значение в DAC0 и DAC1 $84,DAC0H,DAC0L,DAC1H,DAC1L,x,CRC8
; Команда сделать N выборок через время th+tl в микросекундах, Tzh+Tzl.
; $85,N,th,tl,Tzh,Tzl,CRC8
;-----
$INCLUDE (C8051F060.inc)

;-----
; настройка стека
;-----

STACK SEGMENT IDATA ;выбор сегмента стека
RSEG STACK
DS 80h ;резервирование 128 байтов для стека
;-----
; настройка таймеров
;-----

CSEG AT 0
ljmp Main

Temp_2 SEGMENT CODE;
RSEG Temp_2 ;выбор сегмента CODE
USING 0 ;использование нулевого банка регистров

Main:
mov WDTCN, #0deh ;выключение watchdog timer
mov WDTCN, #0adh
mov SP, #STACK-1 ;инициализация указателя стека

mov SFRPAGE,#0FH ;выбор страницы 0fH
MOV XBR2,#40H

MOV P3MDOUT,#0FFH ;настройка портов на вывод
MOV P2MDOUT,#0FFH ;настройка портов на вывод
MOV P5MDOUT,#0FFH ;настройка портов на вывод
MOV P6MDOUT,#0FFH ;настройка портов на вывод

;-----
;основная программа
;-----
G0:
CLR P3.1 ;настройка буфера на передачу данных
MOV P2,#01101101B ;
CLR P3.3 ;разрешение работы буфера
SETB P3.3 ;запрет работы буфера
MOV P2,#10010010B ;
CLR P3.3 ;разрешение работы буфера
JMP G0 ;повтор программы

END

```

При обновлении информации необходимо чтобы период обновления информации $T_{обн}$ был больше пропускной способности оператора.

ПРОГРАММА РАБОТЫ

1. Подключить лабораторный стенд к персональному компьютеру через разъемы для внешних устройств.
2. Записать таблицу преобразования шестнадцатиричных чисел в коды семисегментного индикатора.
3. Составить алгоритм вывода информации на индикаторы.
4. Составить программу вывода информации на индикаторы. Для каждого индикатора выделить одну ячейку ОЗУ для хранения данных.
5. Записать в выделенные ячейки ОЗУ следующие данные:

для первого индикатора	1D
для второго индикатора	2D
для третьего индикатора	3D
для четвертого индикатора	4D
6. Вывести записанную информацию на информационное табло.
7. Снять осциллограммы напряжений на сегментах индикаторов.
8. Определить критическую частоту мелькания изображения.
9. Определить время выборки одного индикатора.
10. Написать программу смены информации на индикаторах.
11. Изменяя частоту управляющего напряжения определить время реакции индикатора.
12. Снять зависимость контраста от частоты управляющего напряжения.
13. Определить пороговое напряжение индикатора.

СОДЕРЖАНИЕ ОТЧЕТА

Отчет должен содержать:

1. Структурную схему блока индикации для двухкоординатного метода управления.
2. Принципиальную схему блока управления четырехразрядным жидкокристаллическим индикатором для двухкоординатного метода управления и поразрядного метода индикации.
3. Принципиальную схему блока управления четырехразрядным жидкокристаллическим индикатором для двухкоординатного метода управления и фазоимпульсного метода индикации.
4. Расчет элементов схемы управления индикаторами для двух методов индикации при $I_{сегм.} = 10 \text{ } \mu\text{A.}$, $U_{пор} = 9 \text{ В.}$
5. Алгоритм и программу вывода информации.

6. Осциллограммы напряжений на разрядах и сегментах индикаторов.
7. Расчет пропускной способности оператора.
8. Зависимость контраста от частоты управления.
9. Расчет порогового напряжения индикатора
10. Выводы.
11. Ответы на контрольные вопросы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. К каким типам индикаторов относится жидкокристаллический индикатор?
2. Как определяется контрастность изображения?
3. Почему для управления жидкокристаллическим индикатором применяется переменное напряжение?
4. Назовите преимущества жидкокристаллических индикаторов по сравнению с полупроводниковыми.
5. Какие свойства жидкокристаллического вещества используют при производстве жидкокристаллических индикаторов?
6. Дайте определение времени включения и времени выключения индикатора.
7. Какой метод индикации применен в лабораторной работе, какие еще методы существуют?
8. Нарисуйте принципиальную схему управления жидкокристаллическими индикаторами для двухкоординатного метода управления.
9. Перечислите основные параметры и характеристики жидкокристаллических индикаторов.
10. Как можно регулировать контрастность изображения на жидкокристаллическом индикаторе?
11. Почему жидкокристаллические индикаторы относятся к пассивным типам индикаторов?

ЛАБОРАТОРНАЯ РАБОТА №4

ИССЛЕДОВАНИЕ СПОСОБОВ ФОРМИРОВАНИЯ ИЗОБРАЖЕНИЯ НА МАТРИЧНЫХ ЭКРАНАХ

Цель работы: изучение поэлементного, построчного и функционального способов формирования изображения на матричном полупроводниковом экране.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ.

Для отображения буквенной и графической информации широко применяют матричные индикаторные панели с большой информационной емкостью. При использовании для этой цели индикаторов с двухкоординатной матричной адресацией кроме падения яркости с ростом скважности серьезным ограничением является и уменьшение времени выборки. С учетом этого для отображения знакографической информации возможно применение:

- монолитных экранов с ограниченной информационной емкостью, в которых развертка производится не более чем по 100-200 позициям;

- экранов изготовленных из субблоков (при выполнении из них непрерывного поля, с одной стороны, необходима стыковка без потери шага, что ограничивает разрешающую способность индикатора, а с другой - субблоки позволяют применять систему многоматричной адресации, что позволяет увеличить время выборки одного элемента отображения и тем самым повысить кажущуюся яркость);

- экранов с запоминанием информации, где, во-первых, не имеет места спад яркости с увеличением числа позиций развертки и, во-вторых, возможна адресация информации с частотой ниже критической частоты мельканий.

В настоящее время выпускаются газоразрядные, жидкокристаллические и полупроводниковые матричные индикаторы. Наиболее широкое распространение получили матричные жидкокристаллические индикаторы, на базе которых производят мониторы. Газоразрядные матричные панели применяют в основном при построении экранов коллективного пользования. Полупроводниковые матричные индикаторы, из-за высокой потребляемой мощности имеют ограниченное применение.

Основные технические характеристики матричного экрана

В лабораторной работе используется матричный индикатор типа TA15-11GWA фирмы KINGBRIGHT, с размером 8x8 точек.

Для управления матричными индикаторами используется метод двухкоординатной адресации (рис.4.1), который позволяет значительно уменьшить число каналов управления. Для вывода информации по всему индикаторному полю необходимо применять динамический режим работы. В динамическом режиме выборка элемента отображения производится одним из следующих способов.

- поэлементным;
- построчным (или по столбцам);
- функциональным.

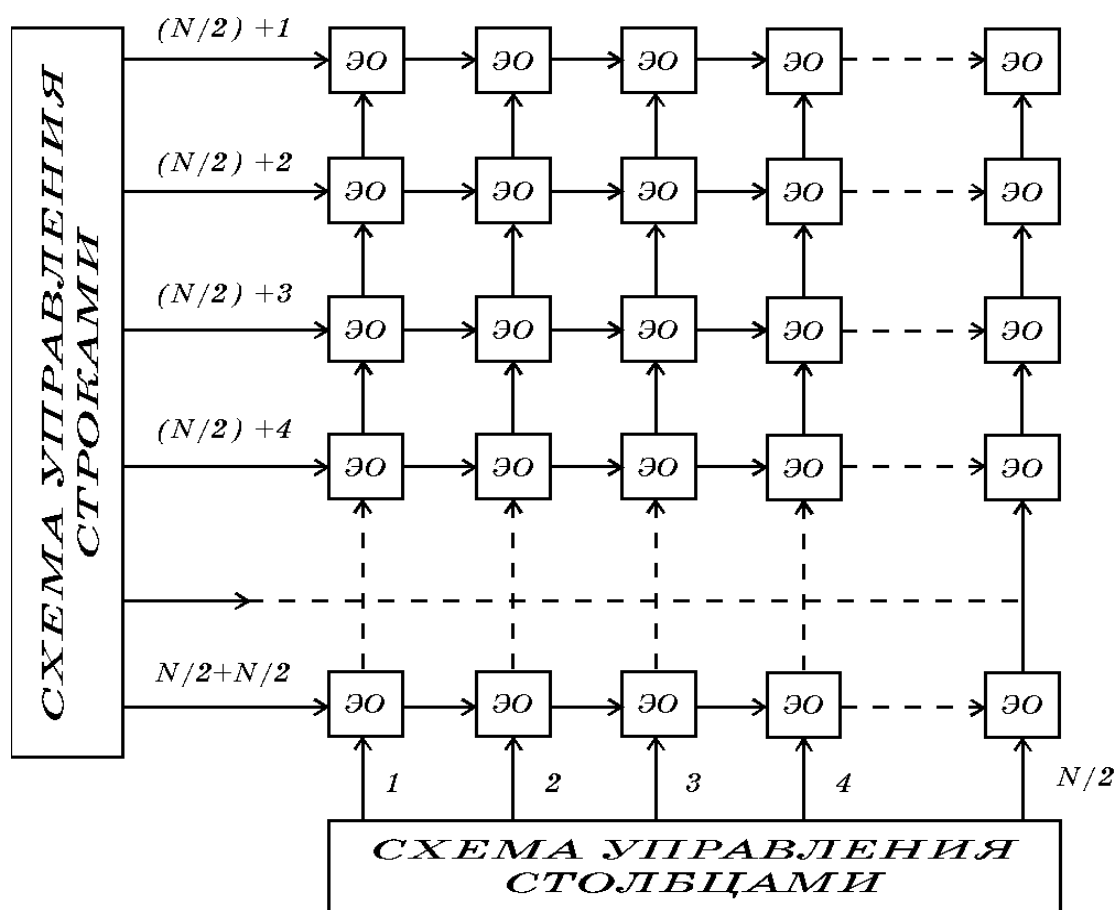


Рис.4.1. Подключение элементов отображения при матричной адресации

В поэлементном способе последовательно производится выборка одного элемента отображения за другим по всему индикаторному полю.

В этом случае:

$$q = N_{стр} \cdot N_{сб} , (4.1)$$

где q - скважность;

$N_{стр}$ - количество строк;

$N_{сб}$ - количество столбцов.

При большом количестве элементов отображения поэлементный способ адресации связан с необходимостью использования мощных импульсных ключей и быстродействующих индикаторов и схем управления ими для получения больших значений мгновенной яркости L_{ν} и малых времен выборки t_{ν} .

При построчном способе последовательно производится выборка одной строки за другой (или столбцов), причем одновременно в каждой строке возбуждаются все необходимые для формирования изображения элементы отображения. Поэтому скважность определяется из выражения:

$$q = N_{стр}, \quad (4.2)$$

где: $N_{стр}$ - количество строк.

В функциональном способе возбуждаются только те элементы отображения, из которых состоит изображение, что дает формулу для определения скважности в виде:

$$q = N_{\phi}, \quad (4.3)$$

где: N_{ϕ} - количество возбуждаемых функциональных элементов отображения.

Из формул (4.1 - 4.3) следует, что наименьшее значение L_{ν} и наибольшее t_{ν} при плотном заполнении экрана информацией обеспечивает способ построчной адресации.

Из изложенного выше следует, что хотя двухкоординатный метод адресации и позволяет значительно уменьшить число линий управления по сравнению с однокоординатным, его существенные недостатки – это уменьшение t_{ν} и L_{ν} при увеличении q . Для уменьшения влияния этих недостатков индикаторное поле разделяют на части и адресуют каждую часть независимо.

ОПИСАНИЕ ЛАБОРАТОРНОЙ УСТАНОВКИ

Лабораторный макет выполнен на базе микропроцессора C8051F060 с двумя матричными индикаторами типа TA15-11GWA (8x8

точек) со схемой управления. Основные параметры матричных индикаторов приведены в таблице 4.1.

Внешний вид и назначение выводов индикатора приведено на рисунке 4.2. Вольт-амперная и кандел-амперная характеристики приведены на рисунке 4.3а и 4.3б соответственно.

Основные параметры индикаторов TX15-11XWA

Таблица 4.1.

Параметр	TX15-11EWA	TX15-11GWA	TX15-11YWA	TX15-11SRWA
Длина волны (нм)	625	568	588	640
Постоянный прямой ток (mA)	30	25	30	30
Импульсный прямой ток (mA)	160	140	140	155
Прямое напряжение (В)	2,0-2,5	2,2-2,5	2,1-2,5	1,85-2,5
Обратный ток (μA)	10	10	10	10
Обратное напряжение (В)	5	5	5	5
Сила света (μcd)	3000-10500	1900-4700	3000-8000	12000-26000
Рассеиваемая мощность мВт	105	105	105	100
Диапазон рабочих температур	-40 - +85	-40 - +85	-40 - +85	-40 - +85

Управление столбцами осуществляется через регистр DD26, а управление строками через регистры DD20 и DD21 рисунок 4.3.

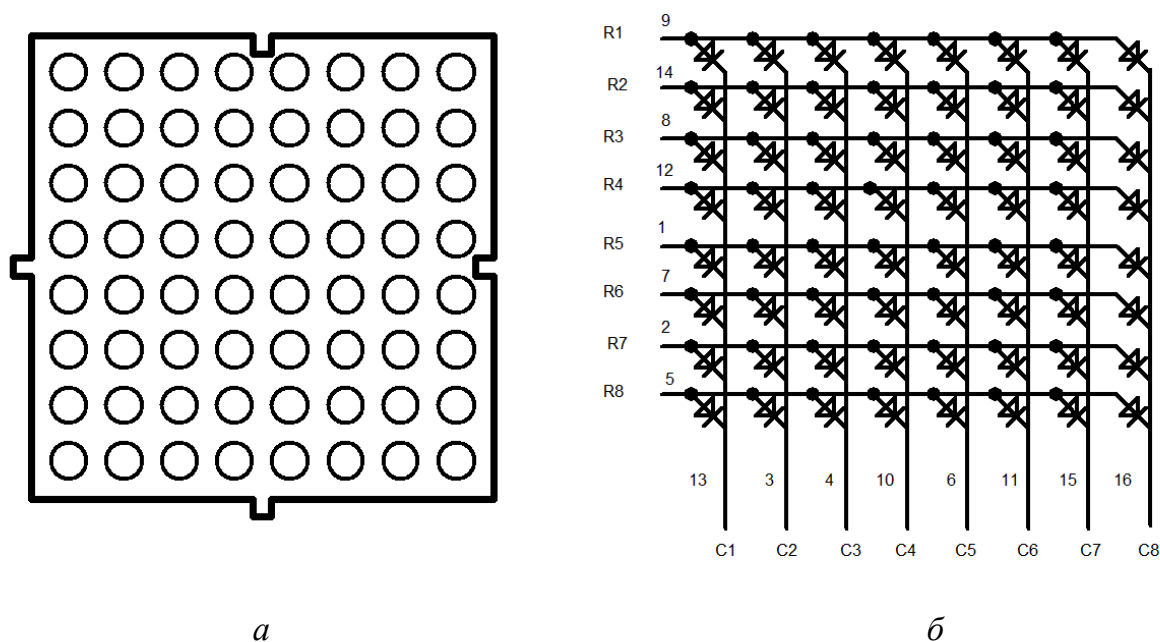


Рис.4.2. Индикатор TA15-11, а - внешний вид, б - назначение выводов

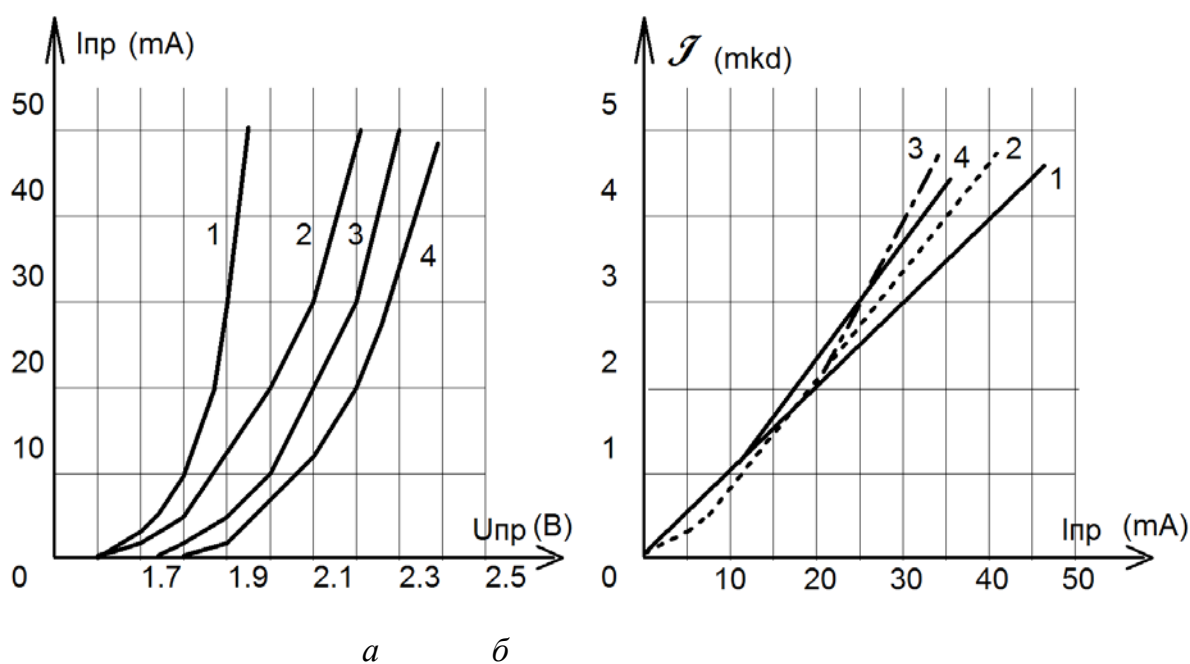


Рис.4.3. Характеристики полупроводниковых матричных индикаторов TA15-11, а – вольт-амперная, б – кандел-амперная, 1 – суперяркий красный, 2 – красный, 3 – зеленый, 4 – желтый

Для увеличения импульсного тока по столбцам матричного экрана применены мощные импульсные ключи, выполненные на транзисторах VT1-VT8. Ограничение импульсного тока на уровне 80 мА

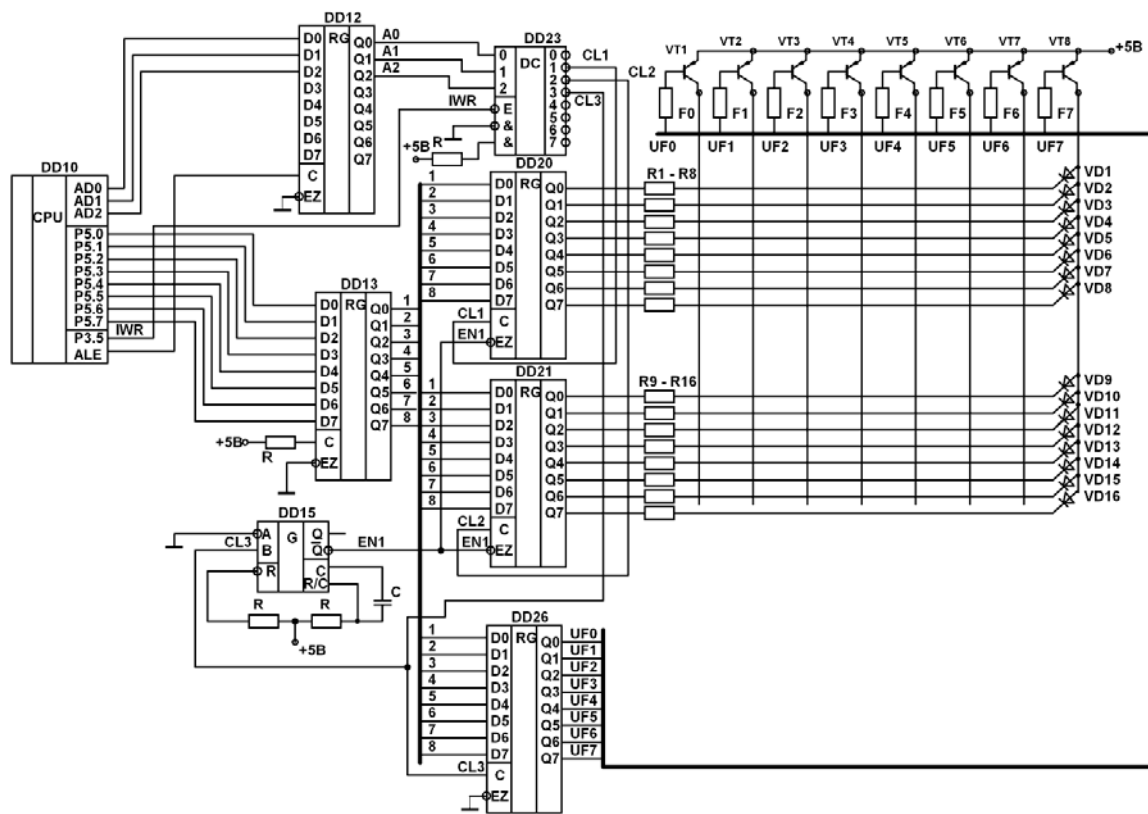


Рис. 4.3. Принципиальная схема управления матричным полупроводниковым индикатором 8Х16 точек

Для возбуждения конкретного элемента отображения необходимо выбрать столбец, в котором находится элемент отображения. Выбор столбца осуществляется подачей уровня логического нуля на соответствующий вывод регистра DD26. Для этого настраиваем порт P5 микроконтроллера на вывод, выводим логический ноль в соответствующий разряд порта P5. Настраиваем порт AD на вывод и выводим в порт AD код 00000011b. По сигналу ALE этот код записывается в регистр DD12 и поступает на вход дешифратора DD23. По сигналу IWR с порта P3 на выходе 3 дешифратора появляется активный уровень, который записывает данные в регистр DD26. После этого производится выбор элемента отображения. Выбор элемента отображения в данном столбце осуществляется подачей уровня логического нуля на соответствующий вывод порта P5. Для записи этих данных в регистр DD20 необходимо установить на выходе порта AD код 00000001b, по сигналу ALE этот код записывается в регистр DD12 и поступает на вход дешифратора DD23. По сигналу IWR с порта P3 на

выходе 1 дешифратора появляется активный уровень, который записывает данные в регистр DD20. Для записи данных в регистр DD21 необходимо установить на выходе порта AD код 00000010b, по сигналу ALE этот код записывается в регистр DD12 и поступает на вход дешифратора DD23. По сигналу IWR с порта P3 на выходе 2 дешифратора появляется активный уровень, который записывает данные в регистр DD21. Перед переходом на следующий столбец индикатора необходимо осуществить гашение элементов отображения выбранного столбца.

На рис. 4.4. приведен алгоритм программы вывода информации на матричный экран. При переходе с одного столбца на другой в программе необходимо предусмотреть гашение сегментов. Частота перебора столбцов определяется из условия равенства критической частоте мельканий.

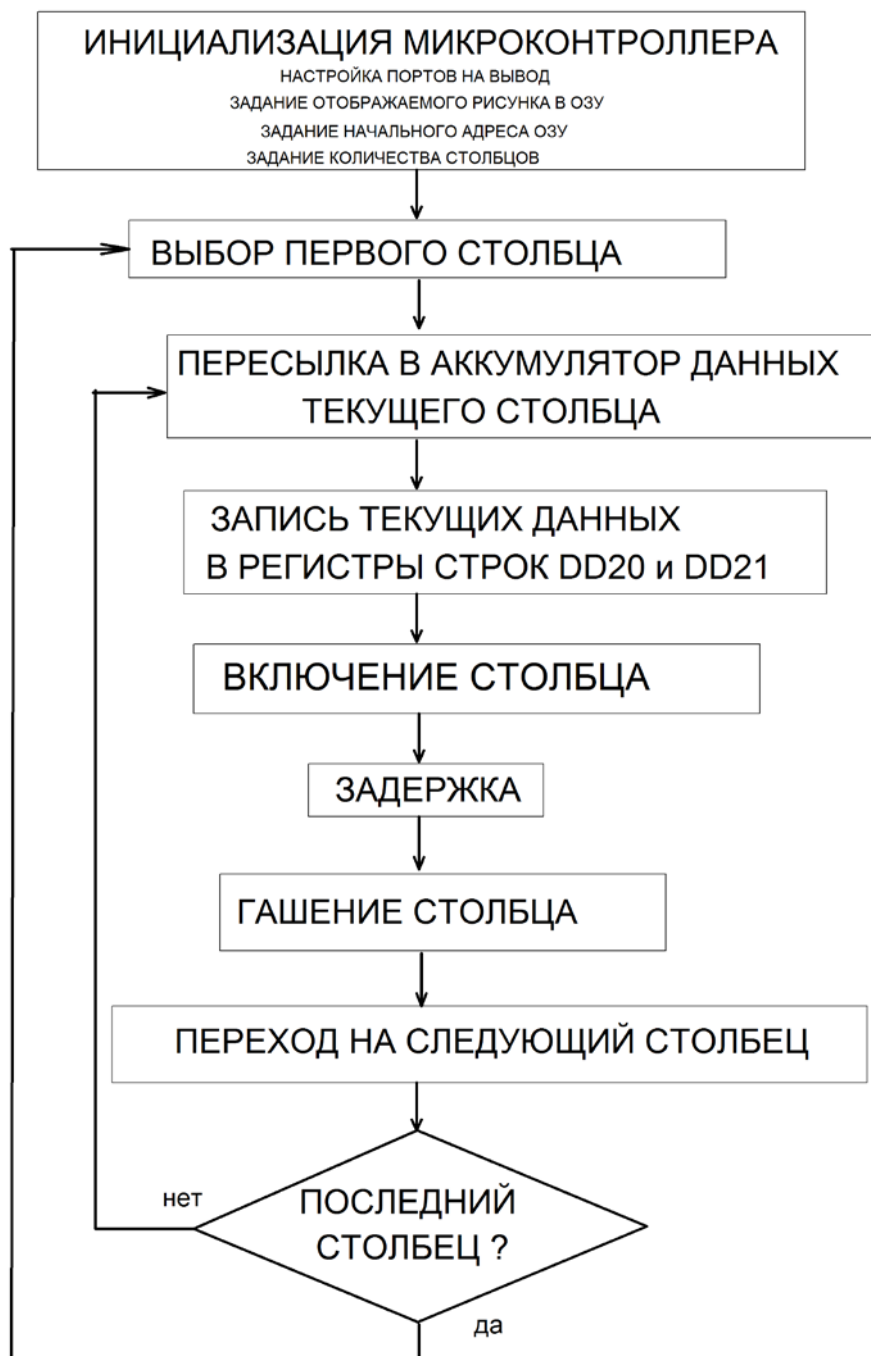


Рис.4.4. Алгоритм программы для управления матричным индикатором

Ниже приведены примеры программы реализующий представленный на рисунке 4.4 алгоритм.

- код программы на языке ассемблера.

; Lab_Test.ASM программа учебно-лабораторног учебного стенда.

```

;-----
; ЗАДАНИЕ РЕГИСТРОВ

;-----
$INCLUDE (C8051F060.inc)
    R9    EQU 0X09;
    R10   EQU 0X0A;
    R11   EQU 0X0B;
    R12   EQU 0X0C;
    R13   EQU 0X0D;
    R14   EQU 0X0E;
    R15   EQU 0X0F;
    R16   EQU 0X10
    R17   EQU 0X11;
    R18   EQU 0X12;
    R19   EQU 0X13;
    R20   EQU 0X14;
    R21   EQU 0X15;
    R22   EQU 0X16;
    R23   EQU 0X17;
    R24   EQU 0X18;

;-----
; STACK          настройка стека;

STACK          SEGMENT IDATA ;выбор сегмента стека
RSEG STACK
DS 80h         ;резервирование 128 байтов для стека

;-----
; настройка таймеров

;-----
    CSEG AT 0
    ljmp Main
;    org 23h
;    ljmp UART_IRQ      ; Обработка прерывания от UART
;    org 2Bh
;    ljmp Timer2_IRQ     ; Вектор обработки прерывания от 2
таймера.

;-----
; MAIN PROGRAM CODE

;-----
Temp_2    SEGMENT CODE ;

```

```
RSEG Temp_2    ;выбор сегмента CODE
USING 0        ;использование нулевого банка регистров
```

Main:

```
mov  WDTCN, #0deh ;выключение watchdog timer
mov  WDTCN, #0adh
mov  SP, #STACK-1 ;инициализация указателя стека
mov  SFRPAGE, #0FH ;выбор страницы 0fH
MOV  XBR2, #40H
MOV  P3MDOUT, #0FFH ;
MOV  P4MDOUT, #0FFH ;
MOV  P5MDOUT, #0FFH ;
MOV  P7MDOUT, #0FFH ;настройка портов на вывод
```

GO1:

```
;-----
;основная программа
;-----
CLR  P6.6 ;выбор первого индикатора
MOV  R0, #09H;
MOV  R1, #0X ;ввод начального адреса рисунка
MOV  R2, #10H ;количество столбцов в обоих индикаторах
MOV  R4, #11111001B ;выбор первого индикатора
MOV  R5, #11111010B ; выбор второго индикатора
MOV  R6, #08H ;количество столбцов в первом индикаторе
MOV  R7, #11111110B ;выбор начального столбца
MOV  R9, #11111110B;
MOV  R10, #11111110B;
MOV  R11, #10000000B;
MOV  R12, #11111110B;
MOV  R13, #11111110B;
MOV  R14, #10000001B;
MOV  R15, #11111101B;
MOV  R16, #11111101B; ЗАДАНИЕ РИСУНКА НА
ИНДИКАТОРАХ
MOV  R17, #11111101B;
MOV  R18, #11111101B;
MOV  R19, #10000001B;
MOV  R20, #11111111B;
MOV  R21, #11011001B;
MOV  R22, #10110111B;
MOV  R23, #10110111B;
```



```

MOV R24,#11000001B;
GO:
;-----
;-----
;ПЕРВЫЙ ИНДИКАТОР
;-----
;-----
MOV A,@R0 ;загрузка элементов рисунка в аккумулятор
;-----
;первая горизонтальная линия
;-----
SETB P3.5 ;
CLR P4.5 ;выключение регистра и дешифратора
MOV P5,A ;загрузка элементов рисунка в порт
JB P6.6,G8 ;если бит установлен, то выбор первого индикатора
MOV P7,R4 ;выбор первого индикатора
JMP G9 ;переход
G8:
MOV P7,R5 ;выбор второго индикатора
G9:
setb P4.5 ;
clr P3.5 ;включение регистра и дешифратора

SETB P3.5 ;
CLR P4.5 ;выключение регистра и дешифратора

mov P5,R7 ;загрузка столбцов
mov P7,#11111011B ;вывод номера столбца
setb P4.5 ;
clr P3.5 ;включение регистра и дешифратора
;-----
;ГАШЕНИЕ
;-----
SETB P3.5 ;
CLR P4.5 ;выключение регистра и дешифратора

mov P5,#0FFH ;гашение всех столбцов
mov P7,#0FFH ;вывод столбцов на индикатор

setb P4.5 ;
clr P3.5 ;включение регистра и дешифратора

```

```

SETB P3.5      ;
CLR P4.5      ;выключение регистра и дешифратора

MOV P5,#0FFH   ;гашение строк
JB  P6.6,G10 ;если бит установлен, то выбор первого
индикатора
MOV P7,R4 ;выбор первого индикатора
JMP G11 ;переход
G10:
MOV P7,R5 ;выбор второго индикатора
G11:
setb P4.5      ;
clr P3.5      ;включение регистра и дешифратора

acall zadergka ;подпрограмма задержки
;-----
;цикл
;-----
G7:
DJNZ R6,G12 ; проверка: все-ли столбцы опрошены
SETB P6.6 ;если да, то установить бит
G12:
MOV A,R7 ;
RL A;
MOV R7,A ;переход к следующему столбцу
INC R0 ; переход к следующей строке
DJNZ R2,GO ;проверка: все ли индикаторы опрошены
JMP GO1 ;цикл
;-----
;подпрограмма задержки
;-----
zadergka:
MOV R3,#0ffH ; задержка 256 циклов
G1: DJNZ R3,G1;
ret

NOP
END

```

- код программы на языке C++.

```
#include "C8051F060.h"
```

```

sbit a=P3^5;
sbit b=P4^5;
void PORT_sett (void);
void work (void);
void preobr (void);
void TIM_sett (void);
void delay (unsigned int del);
void otpravka (k, l);
void massiv (void);
int v_0[]={0x00, 0x7F, 0x7F, 0x00, 0x7F, 0x7F, 0x00, 0xFF};
int v_1[]={0x00, 0x7F, 0x7F, 0x00, 0x7F, 0x7F, 0x00, 0xFF}; // инфа на 1-
ом индикаторе "III"
int v_2[]={0xff, 0x00, 0xe7, 0xdb, 0xbd, 0x7e, 0xff, 0xFF}; // инфа на 2-ом
индикаторе "K"
int i, p, j, c, k, l,s;
void main (void)
{
    SFRPAGE = CONFIG_PAGE;
    EA=0; //
    WDTCN = 0xDE; // выключение watchdog timer
    WDTCN = 0xAD;
    EA=1;
    OSCICN = 0x83; // Настройка f=24.5 MHz
    PORT_sett ();
    while(1)
    {
        j=0xFA; // выбор первой матрицы
        massiv ();
        work ();
        j=0xF9; // выбор второй матрицы
        massiv ();
        work ();}
}
//настройка портов
void PORT_sett (void)
{
    SFRPAGE = CONFIG_PAGE;
    XBR2 = 0x40; //включаю матрицу ввода - вывода
    P3MDOUT=0xFF;

```

```

    P4MDOUT=0xFF;
    P5MDOUT=0xFF;
    P7MDOUT=0xFF;      //настройка портов P3-P5, P7 на вывод
}
//настройка таймера для формирования задержки
void TIM_sett (void)
{
    SFRPAGE = TIMER01_PAGE;
    TMOD |= 0x01;      //для формирования 1 мс
}
void work (void)
{
    i=0;                // выбор первого столбца
    while (i<=7)
    { preobr ();
//вывод на один индикатор
//-----
//Цыфра горит

        k=p;            //информация в i-ом столбце
        l=j;            // выбор DD20
        otpravka (k, l);
        k=c;            //выбор i-го столбца
        l=0xFB;         //выбор DD 26
        otpravka (k, l);
        delay (5);      // формирование задержки
//цифра не горит!!
        k=0xFF;         //тушение светодиодов
        l=j;            // выбор DD20
        otpravka (k, l);
        k=c;            //выбор i-го столбца
        l=0xFB;         //выбор DD 26;
        otpravka (k, l);

//-----
        delay (1);      // задержка

```

```

        i++;                // следующий столбец
    }
}
void preobr (void)
// ввод: p преобразуемый символ, вывод: p- код на семисегментнике
{
    p=v_0[i];
    c=0xFB;
    c=(c<<i)|(c>>(8-i));}
void otpravka (k, l)
{
    a=1;                    //запрет работы дешифратора
    b=0;                    //запрет работы регистра DD12
    P5=k;                   //отправка данных на порт P5
    P7=l;                   //выбор микросхем с помощью дешифратора
    b=1;                    //Разрешение работы регистра DD12
    a=0;                    //Разрешение работы дешифратора
}
//формирователь задержек
void delay (unsigned int del)
{
    SFRPAGE = TIMER01_PAGE;
    //для мс
    TH0 = 0xFE;
    TL0 = 0x0B;
    TR0=1;
    while (del!=0)
    {
        while (!TF0){}
        TH0 = 0xFE;
        TL0 = 0x0B;
        del--;
        TF0=0;}
    TR0=0;
    SFRPAGE = CONFIG_PAGE;
}
void massiv (void)
// перезапись массивов
{
    s=0;
    while (s<=7)

```

```

    {if (j==0xF9)
        {v_0[s]=v_1[s];}
    if (j==0xFA)
        {v_0[s]=v_2[s];}
        s++;}
}

```

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Подключить лабораторный стенд к персональному компьютеру.
2. Составить алгоритм вывода данных из ОЗУ на индикаторы.
3. В ОЗУ сформировать данные для отображения букв.
4. Составить программу вывода информации из ОЗУ на индикаторы
5. Вывести на индикаторы две цифры по заданию преподавателя.
6. Отобразить на экране индикаторов первую и вторую буквы своей фамилии с использованием построчного, поэлементного и функционального способов формирования изображения.
7. Снять осциллограммы напряжений на столбцах и строках индикатора для п.5 и п.6.
8. Осуществить вывод информации в виде последовательного перебора букв своей фамилии.
9. Определить критическую частоту мелькания и пропускную способность оператора.
10. Определить время выборки одного элемента отображения при построчном, функциональном и поэлементном способе формирования изображения.

СОДЕРЖАНИЕ ОТЧЕТА

Отчет должен содержать:

1. Структурную схему блока управления матричным индикатором.
2. Принципиальную схему блока управления полупроводниковым матричным индикатором.
3. Расчет элементов принципиальной схемы управления матричным индикатором.
4. Алгоритм и программу вывода информации.

5. Осциллограммы напряжений на столбцах и строках матричного индикатора.
6. Результаты измерений.
7. Выводы.
8. Ответы на контрольные вопросы.

ВАРИАНТЫ ЗАДАНИЙ

1. Отобразить на матричном экране первую и вторую буквы своей фамилии, используя построчный, поэлементный и функциональный способ возбуждения элементов отображения.
2. Отобразить на матричном экране в режиме бегущей строки номер своей группы.
3. Отобразить на матричном экране графическую информацию:
 - синусоиду;
 - прямоугольный импульс;
 - пилообразный сигнал;
 - треугольный импульс;
 - колоколообразный сигнал;
 - гистограмму;
 - круг.
4. Отобразить на матричном экране мерцающий маркер с секундомером.
5. Отобразить на матричном экране поочередно сменяющуюся графическую информацию:
6. Плавное изменение яркости от столбца к столбцу.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое критическая частота мельканий?
2. Нарисуйте структурную схему устройства отображения информации с двухкоординатным способом подключения элементов отображения.
3. Относится ли матричный способ адресации к двухкоординатному и почему?
4. Как изменится кажущаяся яркость $L_{каж}$ при увеличении скважности, при неизменных других параметрах?
5. Каким образом определяется скважность при поэлементном, построчном и функциональном способе возбуждения элементов отображения, для матричного экрана?

6. Объясните принцип управления матричным ЖК-индикатором.
7. Дайте сравнительную оценку динамических параметров матричных индикаторов разных типов?
8. Как можно регулировать яркость изображения?

ЛАБОРАТОРНАЯ РАБОТА №5

ИЗУЧЕНИЕ ПРИНЦИПОВ ФОРМИРОВАНИЯ ИЗОБРАЖЕНИЯ НА ЭЛЕКТРОННО — ЛУЧЕВЫХ ТРУБКАХ

Цель работы: Ознакомление с основными типами цветных кинескопов и определение их основных характеристик.

1.КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Кинескопом называется оконечное телевизионное устройство, предназначенное для воспроизведения изображения.

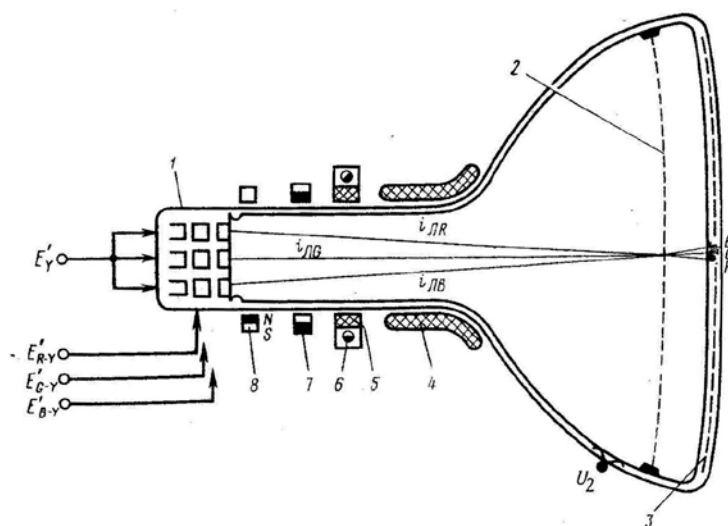


Рис.5.1.Масочный кинескоп

1- электронно-оптическая система, 2-теневая маска, 3-люминофорный экран, 4-строчные отклоняющие катушки, 5-кадровые отклоняющие катушки, 6 и 7-магниты чистоты цвета, 8-магнит горизонтального перемещения, E_y -катоды, E_{R-Y} , E_{G-Y} , E_{B-Y} -модуляторы.

Изображение на экране кинескопа формируется при облучении люминофора пучком электронов. Кинескоп цветного изображения содержат люминофор трех основных цветов и три электронных пушки (ЭП), формирующие три электронных луча с независимым управлением интенсивностью каждого луча (рис.5.1).

Существуют цветные кинескопы с одним электронным лучом, так называемые хроматроны. Однако они не получили распространения из-за сложности схем управления.

1.1. Особенности конструкции кинескопов и вносимые ими искажения.

На рисунках 5.2 и 5.3 показано различие в расположении ЭП (а), устройстве теневой маски (б) и расположении зерен люминофора (в) у кинескопов с дельтаобразным и планарным расположением ЭП.

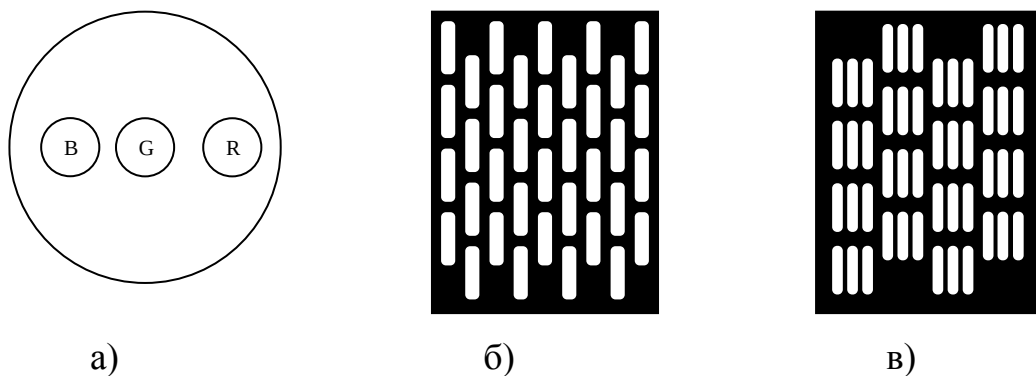


Рис. 5.2. Кинескоп с планарным расположением ЭП

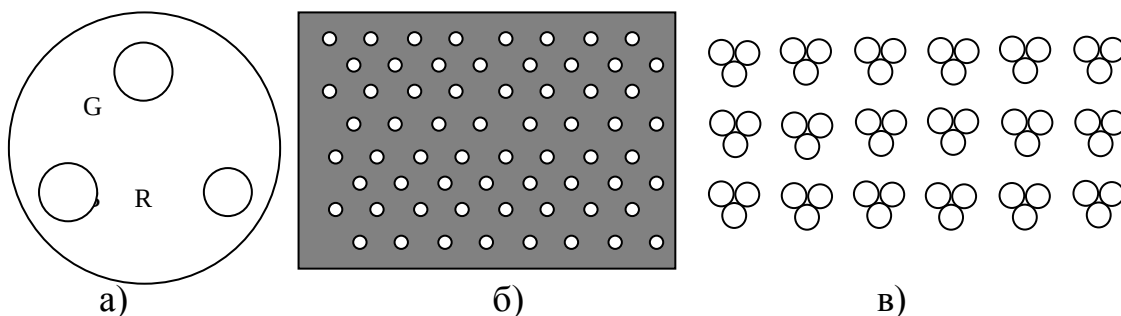


Рис. 5.3. Кинескоп с дельтаобразным расположением ЭП

В кинескопах с дельтаобразным расположением ЭП они размещаются по углам равностороннего треугольника, а с планарным — в одной плоскости, причем в центре (на горизонтальной оси кинескопа) находится ЭП зеленого, а по обеим сторонам от него — красного и синего цветов (рис. 5.2а). В кинескопах с дельтаобразным расположением ЭП применен мозаичный экран, в котором в определенной последовательности нанесены зерна люминофора красного, синего и зеленого цветов, объединенные в триады. В кинескопах с планарным расположением ЭП люминофоры нанесены в виде чередующихся полосок красного, зеленого и синего цветов. Так как размер зерен люминофора очень мал (диаметр 0,3 мм), то с

расстояния, на котором производится просмотр телевизионных изображений, разрешающая способность глаза не позволяет различить их отдельно друг от друга. В результате свечение зерен люминофорной триады или расположенных рядом каждой из нанесенных на экраны чередующихся полосок люминофоров красного, зеленого и синего цветов воспринимается человеческим глазом как суммарное. Происходит пространственное аддитивное сложение, при котором яркость, насыщенность и цветовой тон результирующего цвета определяются суммарной энергией и соотношением составляющих излучения всех электронных лучей. Для того чтобы каждый из электронных лучей при движении в процессе развертки возбуждал люминофорные зерна или полосы связанного с ним цвета, применяют теневую маску, располагаемую в непосредственной близости от экрана. Маска выполняется из стального листа толщиной 0.15 мм. В кинескопах с дельтаобразным расположением ЭП такая маска имеет круглые, а в кинескопах с планарным расположением ЭП - продолговатые отверстия (щели). Центр каждого отверстия маски диаметром 0,2.. 0.3 мм в кинескопах с дельтаобразным расположением ЭП находится против центра одной триады люминофоров, а в кинескопах с планарным расположением ЭП - против зеленой полосы люминофора. Пересекаясь в отверстиях маски, электронные лучи слегка расходятся и попадают каждый на соответствующие зерна или полосы люминофоров. Теневая маска задерживает до 70% электронов, испускаемых катодами. В 1975 году фирмой Sony был представлен кинескоп типа Trinitron с так называемой апертурной сеткой. В нем разделение лучей выполняется не маской, а вертикальными проволочками, натянутыми на месте маски перед люминофором. Форма экрана в таком кинескопе не сферическая, а цилиндрическая. Поглощение электронов сеткой не превышает 30%, поэтому кинескопы тринитрон имеют значительно более высокую яркость и контрастность в сравнении с другими типами.

Свойственные масочным кинескопам искажения раstra можно разделить на подушкообразные и трапецеидальные. Подушкообразные искажения присущи кинескопам с большим углом отклонения и относительно плоским экраном (форма которого отличается от сферической). Радиус кривизны экрана в них значительно больше, чем расстояние от центра отклонения до экрана. По этой причине путь электронного луча до центра экрана, оказывается меньше, чем до края. Трапецеидальные искажения возникают из-за различного расположения прожекторов по отношению к оси кинескопа. При этом каждый из них создает свой растр, отличный по характеру искажений и смещенный по отношению к двум другим.

1.2.Сведение лучей.

Сведением лучей называется совмещение трех цветовых растров. Статическим сведением называется совмещение лучей в центре экрана. При изготовлении кинескопа невозможно выполнить все его элементы настолько точно, чтобы лучи сходились в центре экрана. Кроме того, магнитное поле Земли также влияет на траекторию луча, отклоняя его в вертикальном направлении, причем это отклонение зависит от ориентации кинескопа относительно сторон света. Для кинескопов с дельтаобразной маской это приводит к отклонению лучей от центров отверстий теневой маски и зерен люминофора. Для кинескопов со щелевой маской это отклонение не имеет значения, так как отклоненные по вертикали лучи продолжают засвечивать свои полосы люминофора. Тем не менее для устранения погрешностей изготовления кинескопа необходимо применять устройства статического сведения.

Динамическое сведение – это совмещение лучей на краях экрана. Таким образом, для достижения динамического сведения необходимо устранить геометрические искажения растров. Эта задача довольно трудна и на практике полное сведение на краях экрана не достигается. При этом статическое сведение во всех случаях должно быть полным. Наилучшее сведение лучей достигается в кинескопах типа тринитрон.

Сведение лучей осуществляется проще в кинескопах со щелевой маской. Это одна из причин того, что этот тип кинескопов вытеснил дельтаобразные, по крайней мере, в бытовой телевизионной аппаратуре.

1.3.Координатные искажения телевизионного изображения

Координатные искажения — это нарушения масштаба передачи, которые влекут за собой нарушение геометрического подобия изображения объекту. Геометрическое подобие может нарушаться как при изменении формы раstra на передаче или на приеме, так и при изменении масштаба внутри строки или кадра, вызываемом неравномерностью скорости $V(t)$ развертки (или отклонением закона скорости) за время строки или кадра. Удобнее оценить первые (формы раstra) и вторые искажения отдельно, причем первые иногда называют *геометрическими искажениями*, а вторые — *нелинейными искажениями*. Такое деление весьма условно, так как оба эти искажения нарушают подобие деталей, т. е. приводят к дифференциальным искажениям масштаба, но оно делает более определенной оценку искажений. Искажения формы раstra оцениваются коэффициентом геометрических искажений.

Для оценки дисторсии бочкообразного или подушкообразного вида (рис. 5.4а и 5.4б), которая возникает или из-за дисторсии электронно - оптических систем, или из-за сферической формы экрана приемной трубки, рассчитывают коэффициент геометрических искажений:

$$k_r = \frac{\Delta h}{b} \cdot 100\%$$

$$k_r = \frac{\Delta b}{h} \cdot 100\%$$

Для оценки трапецидальных искажений раstra (рис. 5.4б), возникающих при нарушении ортогональности оптической или электронной осей к плоскости изображения, определяют коэффициент трапецидальных искажений:

$$k_{r,т} = 2 \frac{l_2 - l_1}{l_2 + l_1} \cdot 100\%$$

Для оценки искажения раstra типа «параллелограмм» (рис. 5.4г) возникающих при нарушении ортогональности отклоняющих полей по строке и по кадру, определяют коэффициент искажений:

$$k_{r,п} = 2 \frac{D_2 - D_1}{D_2 + D_1} \cdot 100\%$$

Обозначенные в формулах величины ясны из соответствующих рисунков (5.4а—5.4г). Нарушение формата раstra, т. е. когда номинальное значение b_u/h_u не равно b/h действительному, также приводит к геометрическим искажениям (см. рис. 5.5г, 5.5д). Геометрические искажения возникают и при воздействии на отклоняющие поля низкочастотных периодических помех (рис. 5.4 д).

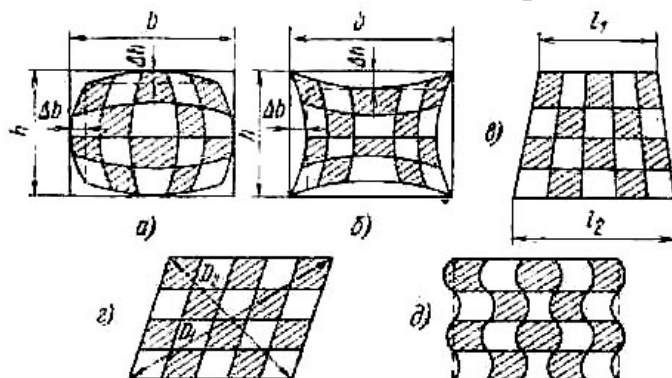


Рис. 5.4. Искажения формы раstra.

Нелинейные искажения могут быть определены через коэффициент нелинейности разверток:

$$K_n = 2 \frac{V_{\max} - V_{\min}}{V_{\max} + V_{\min}} \cdot 100\%$$

где $V_{\max}$ — максимальная скорость движения развертывающего луча по экрану за период строки или кадра; $V_{\min}$ — минимальная скорость движения луча соответственно.

Непосредственно по изображению нелинейные искажения могут быть оценены по искажениям воспроизведения равномерного шахматного поля (рис. 5.5 а):

$$K_n = 2 \frac{d_{\max} - d_{\min}}{d_{\max} + d_{\min}} \cdot 100\%$$

Смысл и значения входящих в формулу величин ясны из рис. 5.5б, 5.5в.

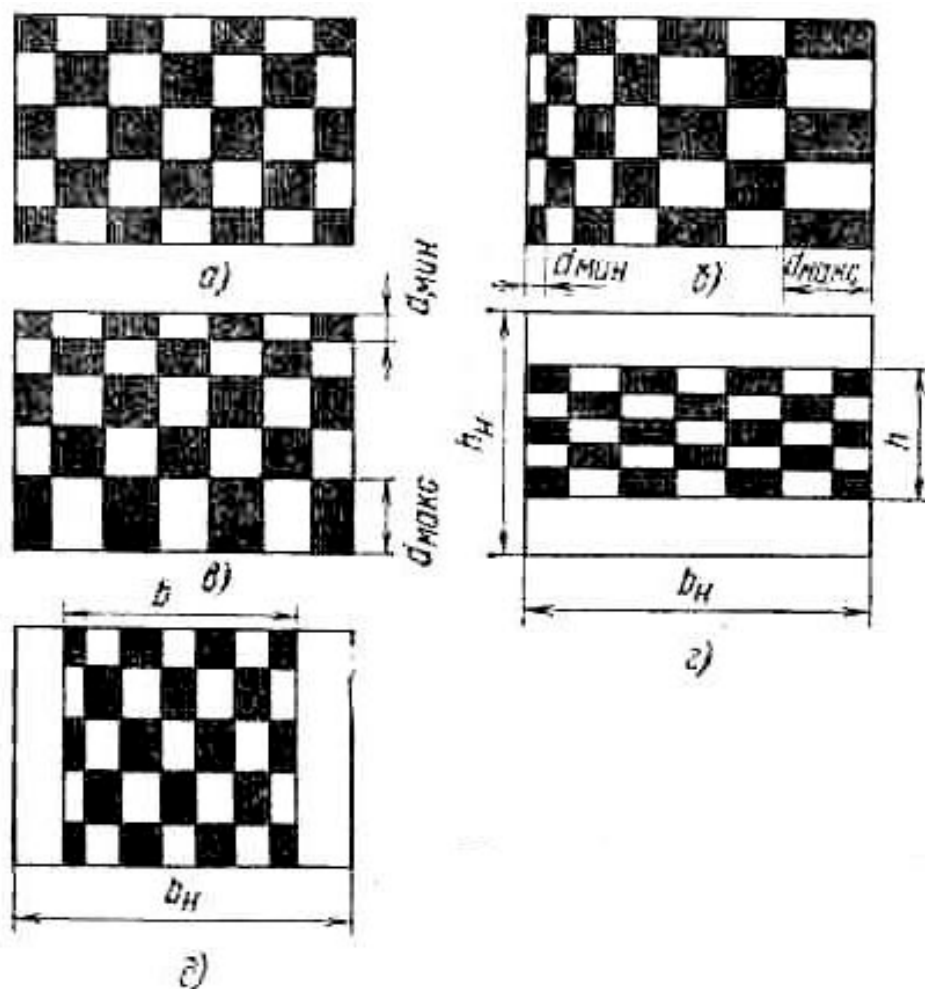


Рис.5.5.Геометрические искажения шахматного поля.

Геометрическое подобие может быть нарушено и другими причинами, зависящими от содержания изображения. На рис. 5.6 приведено изображение того же шахматного поля.

Наблюдаемая на изображении дисперсия отдельных клеток является результатом влияния граничных полей потенциального рельефа мишени передающей трубки на коммутирующий луч с медленными электронами. Здесь модуляционные явления (потенциальный рельеф) повлияли на развертку и создали дифференциальные растровые искажения, т. е. вызвали нарушение геометрического подобия.

Нелинейность развертки до 5% практически незаметна глазом, и изображение может быть определено как удовлетворительное при нелинейности до 15%. Дифференциальные геометрические искажения нормировать весьма трудно, и допустимая их величина в значительной степени зависит от характера самого изображения.

Определенные выше коэффициенты искажения геометрии раstra могут иметь разные знаки. Нормировать удобно суммарные геометрические искажения, которые могут быть измерены по шахматному полю аналогично коэффициенту k_i через значения $d_{\max}$ и $d_{\min}$ безотносительно их места на растре.



Рис.5.6.Геометрические искажения изображения, вызванные особенностью работы передающей трубки типа суперортискон.

2.Состав оборудования

Лабораторная работа выполняется на персональном компьютере с использованием оригинального программного обеспечения и специализированного прибора для бесконтактного определения частоты кадровой и строчной разверток и яркости кинескопа. Инструкция по эксплуатации прибора находится у преподавателя.

3.Порядок работы.

1. Написать программу вывода на экран монитора шахматного поля.
2. Проанализировать изображение на экране. Измерить геометрические искажения в центре и на краях экрана, подушкообразные и трапецеидальные искажения, нелинейность и центровку раstra. Попытаться максимально уменьшить искажения ручками регулировок. Сделать вывод о величине искажений и качестве отклоняющей системы кинескопа.
3. Измерить величину «зерна» и разрешающую способность монитора.
4. Определить максимальную частоту кадровой развертки.
5. Определить максимальную частоту строчной развертки.
6. Написать программу вывода горизонтальной строки вверху экрана монитора и определить качество динамического сведения лучей.
7. Написать программу вывода горизонтальной строки в центре экрана монитора и определить качество статического сведения лучей.
8. Написать программу поочередного отображения одного из основных цветов (R, G, B). Визуально оценить чистоту цветов. Сделать вывод о качестве цветоделительной маски.
9. Написать программу вывода точки в центр экрана. Определить центровку раstra.
10. Определить максимальное разрешение монитора. Для этого написать программу вывода вертикальных чередующихся полос белого и черного цветов через один пиксел. Повторить для горизонтальных полос.

4.СОДЕРЖАНИЕ ОТЧЕТА

Отчет должен содержать:

1. Структурную схему монитора.

2. Результаты проведенных измерений.
3. Расчет частоты строчной развертки и полосы пропускания видеоусилителя.
4. Диаграммы токов в отклоняющих катушках кадровой и строчной разверток для двух режимов работы.
5. Ответы на контрольные вопросы.

5 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем отличие масочного кинескопа от «Хроматрона»?
2. Объясните причину возникновения геометрических искажений в кинескопе?
3. Что такое разрешающая способность?
4. Перечислите используемые в настоящее время системы цветного телевидения.
5. Какая система цветного телевидения применяется в России?
6. Как выбирается соотношение сторон кинескопа?
7. Как определить полосу пропускания видеоусилителя?
8. Зачем производится гашение луча при обратном ходе?
9. Нарисуйте траекторию движения луча по экрану кинескопа при прогрессивной и чересстрочной развертках.
10. Как изменится диаграмма отклоняющего тока в строчных катушках при смене режима работы монитора?

6 ЛИТЕРАТУРА

1. Телевидение / Под ред. проф. П.В. Шмакова. - М.: -Связь, 1979. – 432 с.
2. Хоган Т. Аппаратные и программные средства персональных компьютеров. Справочник в двух томах. - М.: Радио и связь. -1995. - Том 2. – 260 с.
3. Мягеев А.А., Степанов В.Н. Персональные ЭВМ и микро-ЭВМ.-Радио и связь, 1991. – 171 с.

ЛАБОРАТОРНАЯ РАБОТА №6

ИССЛЕДОВАНИЕ МАТРИЧНОГО ЖИДКОКРИСТАЛЛИЧЕСКОГО ИНДИКАТОРА С ВСТРОЕННОЙ СХЕМОЙ УПРАВЛЕНИЯ

Цель работы: изучение принципов формирования изображения на матричных жидкокристаллических индикаторах.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

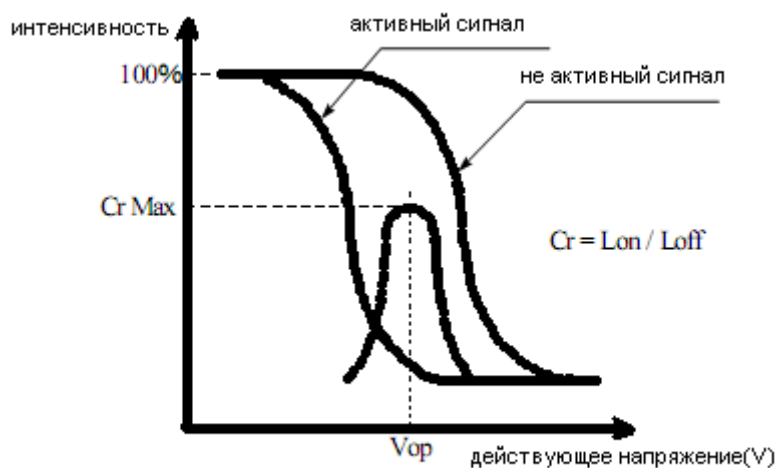
В противоположность активным приборам жидкокристаллические (ЖК) индикаторы не генерируют свет, а только управляют его прохождением, что обуславливает чрезвычайно малую потребляемую ими мощность. Преимуществом ЖК-индикаторов является также малое управляющее напряжение, позволяющее непосредственно согласовывать их с цифровыми интегральными схемами. Так как ЖК-индикаторы относятся к классу пассивных, то основным их оптическим параметром является не яркость, а контрастность (для просветных индикаторов вместо контрастности часто пользуются коэффициентом пропускания, который определяют как отношение интенсивностей выходящего света к падающему).

Вольт-контрастная характеристика жидкокристаллического индикатора зависит не от амплитудного, а от действующего значения приложенного напряжения. Типичные вольт-контрастные характеристики используемого ЖК-индикатора показаны на рис. 6.1.

Динамические параметры ЖК-индикаторов определяются временами реакции τ_{pk} и релаксации τ_{pl} . (рис.6.2). Кривая изменения контраста при наложении импульса напряжения имеет следующие участки:

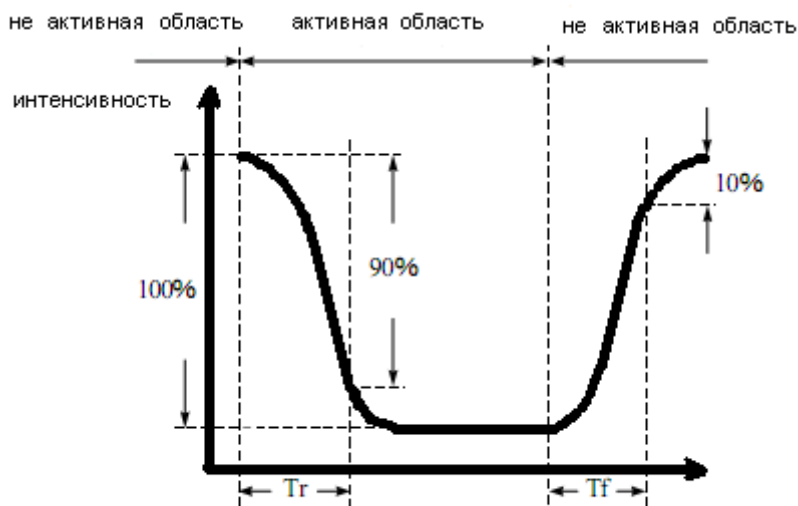
- задержку включения,
- время нарастания (в сумме они равны времени реакции),
- задержку выключения (обычно очень малую),
- время спада (две последние составляющие в сумме дают время релаксации).

Угол обзора индикатора во взаимно-перпендикулярных направлениях показан на рис.6.3.



[положительный тип]

Рис.6.1. Вольт-контрастная характеристика



[положительный тип]

Рис.6.2. Динамические параметры индикатора

Условия для определения напряжения и времени:

- Рабочее напряжение: V_{op} ;
- Частота кадров: 64 Гц;
- Угол обзора(θ , φ): $0^{\circ}C$, $0^{\circ}C$.

ОПИСАНИЕ ЛАБОРАТОРНОЙ УСТАНОВКИ

В лабораторной работе используется стенд на базе микропроцессорного комплекта, обеспечивающего передачу двоичных

данных на матричный жидкокристаллический индикатор WH1604A – YGH – СТ с встроенной схемой управления рис 6.4. Его основные геометрические параметры приведены в таблице 6.1. Предельно допустимые параметры приведены в таблице 6.2. Электрические параметры – в таблице 6.3, а оптические и динамические – в таблице 6.4. Назначение контактов соединительного разъема показано в таблице 6.5.

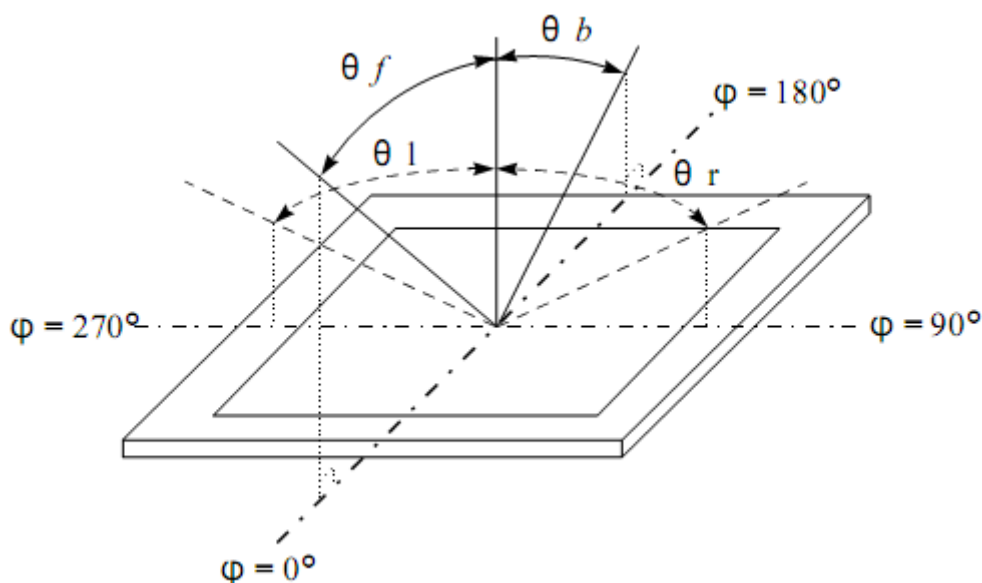


Рис.6.3. Угол обзора индикатора

Технические характеристики ЖК модуля WH1604A – YGH – СТ.

Таблица 6.1.

Наименование параметра	Величина	Единица измерения
Число знаков	16 знаков * 4 линии	-
Размеры модуля	87 * 60 * 13.6(максимум)	мм
Видимая область	62 * 26	мм
Активная область	56.2 * 20.8	мм
Размер точки	0.55 * 0.55	мм
Расстояние между точками	0.6 * 0.6	мм
Размер символа	2.95 * 4.75	мм
Расстояние между символами	3.55 * 5.35	мм
Режим работы	1/16	-
Время работы	6	час
Тип свечения	Желто-зеленое	-

Предельно допустимые параметры

Таблица 6.2.

Наименование параметра	Символ	min	max	Единица измерения
Рабочая температура	Top	-20	+70	°C
Температура хранения	Tst	-30	+80	°C
Входное напряжение	Vi	Vss	Vdd	В
Напряжение питания для логики	Vdd -Vss	-0.3	7	В
Напряжение питания для LCD	Vdd -Vo	-0.3	13	В

Электрические параметры

Таблица 6.3.

Наименование параметра	Обозн.	Условие	min	typ	max	Ед. изм.
Напряжение питания для логики	Vdd- Vss	-	4.5	5	5.5	В
Напряжение питания для LCD	Vdd- Vo	Ta=20°C Ta=25°C Ta=70°C	- - 3.8	- 4.5 -	5.7 - -	В В В
Входное напряжение высокого уровня	Vih	-	0.7 Vdd	-	Vdd	В
Входное напряжение низкого уровня	Vil	-	-	-	0.6	В
Выходное напряжение высокого уровня	Voh	-	3.9	-	-	В
Выходное напряжение низкого уровня	Vol	-	-	-	0.4	В
Ток потребления	Idd	Vdd=5В	1.2	1.5	2	мА

Оптические и динамические параметры

Таблица 6.4

Наименование параметра	Обозн.	Условие	min	typ	max	Ед. изм.
Угол обзора	(V) θ (H) φ	CR $\geq$ 2 CR $\geq$ 2	20 -30	- -	40 30	градус градус
Коэффициент контраста	CR	-	-	3	-	-
Время срабатывания	T rise T fall	- -	- -	150 150	200 200	мс мс

Назначение контактов разъема индикатора

Таблица 6.5

№	Символ	Уровень	примечание
1	Vss	0 В	Общий провод
2	Vdd	5 В	Напряжение питания для логики
3	Vo	Изменяемое	Напряжение питания для LCD
4	RS	H/L	H: данные, L: код инструкции
5	R/W	H/L	H: чтение из модуля, L: запись в модуль
6	E	H, H $\rightarrow$ L	Сигнал включения модуля
7	DB0	H/L	Линия данных
8	DB1	H/L	Линия данных
9	DB2	H/L	Линия данных
10	DB3	H/L	Линия данных
11	DB4	H/L	Линия данных
12	DB5	H/L	Линия данных
13	DB6	H/L	Линия данных
14	DB7	H/L	Линия данных
15	A	-	Регулировка контраста
16	K	-	-

Описание функций

ЖК дисплей построен на LSI контроллере, который имеет два 8-разрядных регистра:

- регистр инструкций (IR)
- регистр данных (DR).

Регистр инструкций содержит коды, которые очищают дисплей и передвигают курсор, а так же он содержит адресную информацию для

памяти данных дисплея(DDRAM) и для генератора символов(CGRAM). Регистр инструкций может только получать информацию от микроконтроллера (MPU). Регистр данных временно хранит данные, которые будут записаны или прочитаны от памяти данных или генератора символов. Когда адресная информация записана в регистр инструкций, тогда данные сохранены в регистре данных из памяти данных дисплея (DDRAM) или из генератора символов (CGRAM). Этими двумя регистрами можно управлять через регистр выбора RS таблица 6.6.

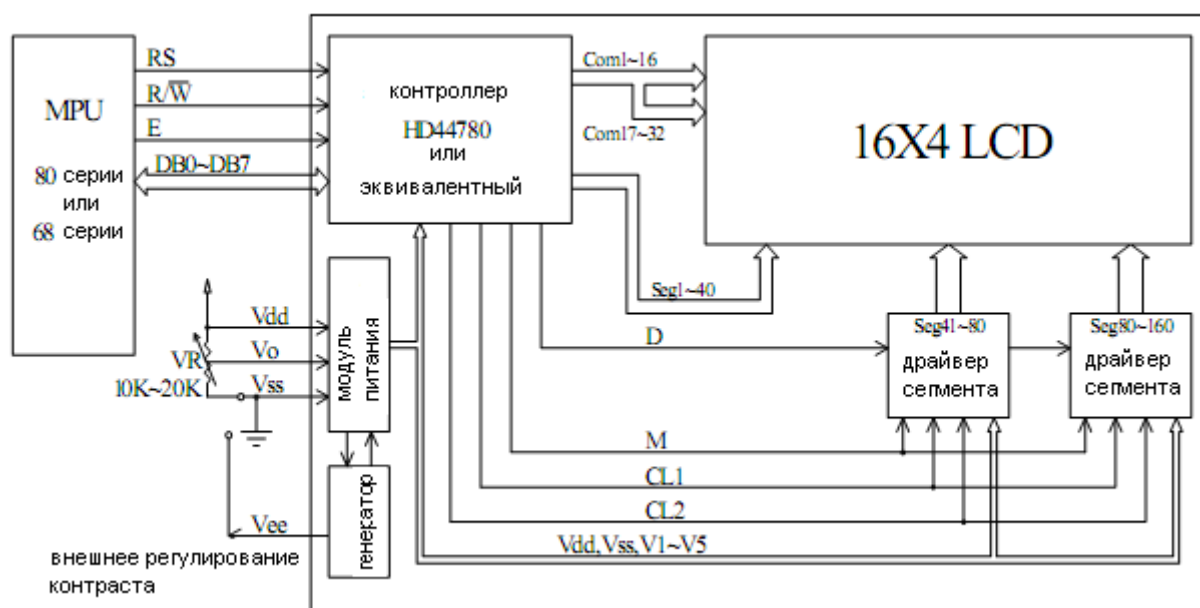


Рис.6.4. Архитектура модуля жидкокристаллического индикатора.

Флаг занятости (BF)

Когда этот флаг равен 1, LSI контроллер находится во внутреннем режиме операций, и следующая команда не будет принята. Когда RS=0 и R/W=1, флаг занятости выводится к DB7. Следующая инструкция может быть записана после выполнения предыдущей команды, когда флаг занятости=0.

Регистр выбора RS

Таблица 6.6.

RS	R/W	Операции
0	0	Управление регистром инструкций (очистка дисплея и т.д.)
0	1	Чтение флага занятости (DB7) и счетчика адреса (DB0-DB7)

1	0	Запись данных в DDRAM или в CGRAM из регистра данных (DR)
1	1	Чтение данных из DDRAM и из CGRAM в регистр данных (DR)

Счетчик адреса (AC)

Счетчик адреса назначает адреса для памяти данных дисплея (DDRAM) и для генератора символов (CGRAM) рис. 6.5, и рис.6.6.

Оперативная память данных дисплея (DDRAM)

Память данных дисплея используется, чтобы хранить данные дисплея, представленные в 8-разрядных символьных кодах рис. 6.5. Его размерность 80*8 бит или 80 символов. В таблице 6.7 показано отношение между адресами DDRAM и нахождением курсора на дисплее.



Рис.6.5. Биты счетчика адреса (AC)



Рис. 6.6. Пример адреса DDRAM - 4E

DDRAM адрес позиции курсора (4 линии по 16 символов)

Таблица 6.7

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F

Генератор символов (CGROM)

Этот генератор генерирует 5*8 и 5*10 точечных символьных образцов соответствующих 8-разрядным символьным кодам. Их соотношения показаны в таблице 6.8.

Кодировка символов знакогенератора

Таблица 6.8

Старшие 4бита Младшие 4бита	0000	00 01	00 10	00 11	01 00	01 01	01 10	01 11	10 00	10 01	10 10	10 11	11 00	11 01	11 10	11 11
0000	CGRAM(1)			0	∂	P	`	p			Б	Ю	ч	,	Д	¼
0001	CGRAM(2)		!	1	A	Q	a	q			Г	Я	ш		Ц	⅓
0010	CGRAM(3)		"	2	B	R	b	r			Ё	б	ъ		Щ	½
0011	CGRAM(4)		#	3	C	S	c	s			Ж	в	ы		д	¾
0100	CGRAM(5)		\$	4	D	T	d	t			З	г	ь		ф	Н
0101	CGRAM(6)		%	5	E	U	e	u			И	ё	э	Х	ц	
0110	CGRAM(7)		&	6	F	V	f	v			Й	ж	ю		щ	
0111	CGRAM(8)		'	7	G	W	g	w			Л	з	я		´	
1000	CGRAM(1)		(	8	H	X	h	x			П	и	«		”	
1001	CGRAM(2)		)	9	I	Y	i	y			У	й	»		~	
1010	CGRAM(3)		*	:	J	Z	j	z			Ф	к	„			
1011	CGRAM(4)		+	;	K	[	k	¹ □			Ч	л	”			
1100	CGRAM(5)		,	<	L	¥	l	¹²			Ш	м	№			
1101	CGRAM(6)		-	=	M	]	m	¹ 5			Ъ	н	¿			
1110	CGRAM(7)		.	>	N	^	n				Ы	п	∫			
1111	CGRAM(8)		/	?	O	_	o				Э	т	£			

Данные для матричного индикатора выставляются на порт P2 и через буферный усилитель DD14 поступают в параллельном виде в контроллер индикатора. Ввод данных осуществляется путем подачи сигнала WRITE, который формируется путем установки 1 в порт P3.1. Выбор регистра управления индикатора осуществляется путем установки 1 в порт P3.0. Для настройки буферного устройства на передачу данных от порта P2 к индикатору необходимо установить на выходе порта P3.1 уровень логической единицы. Для разрешения работы буферного устройства необходимо установить уровень логического нуля на выходе порта P3.3.

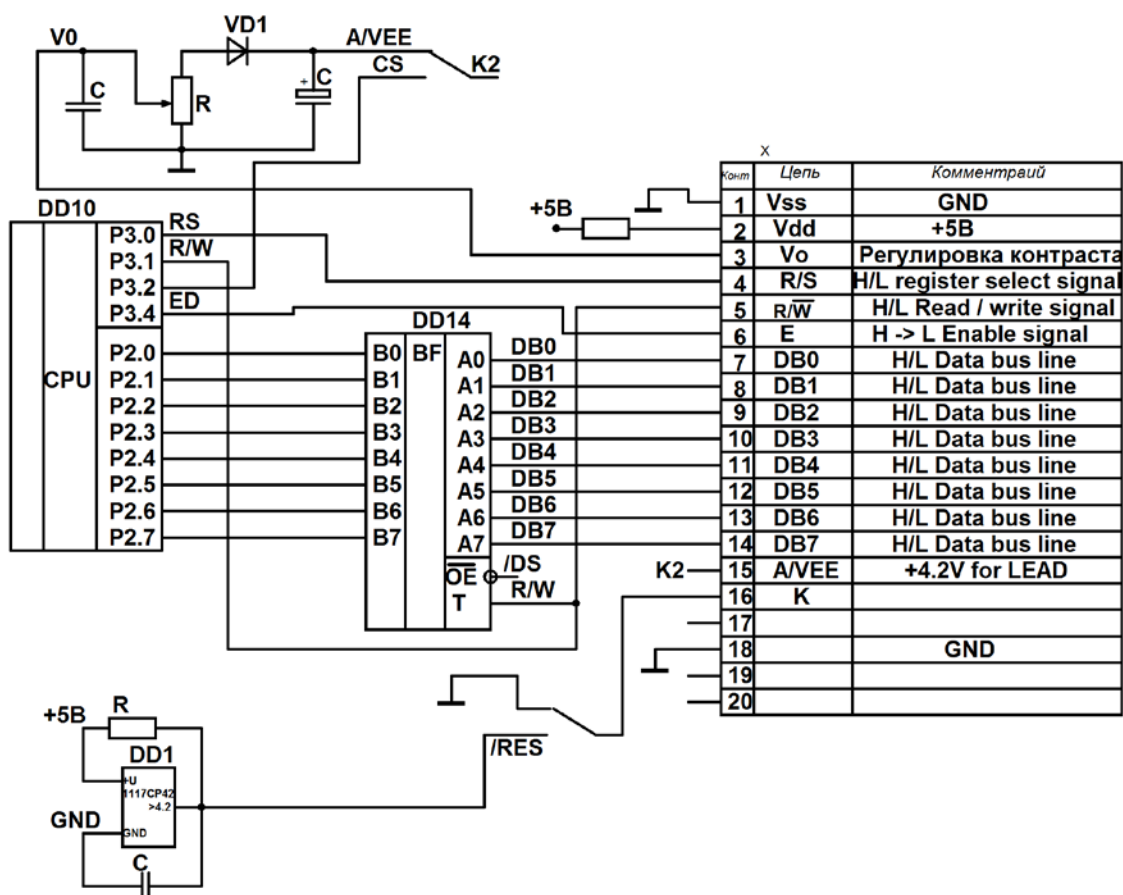
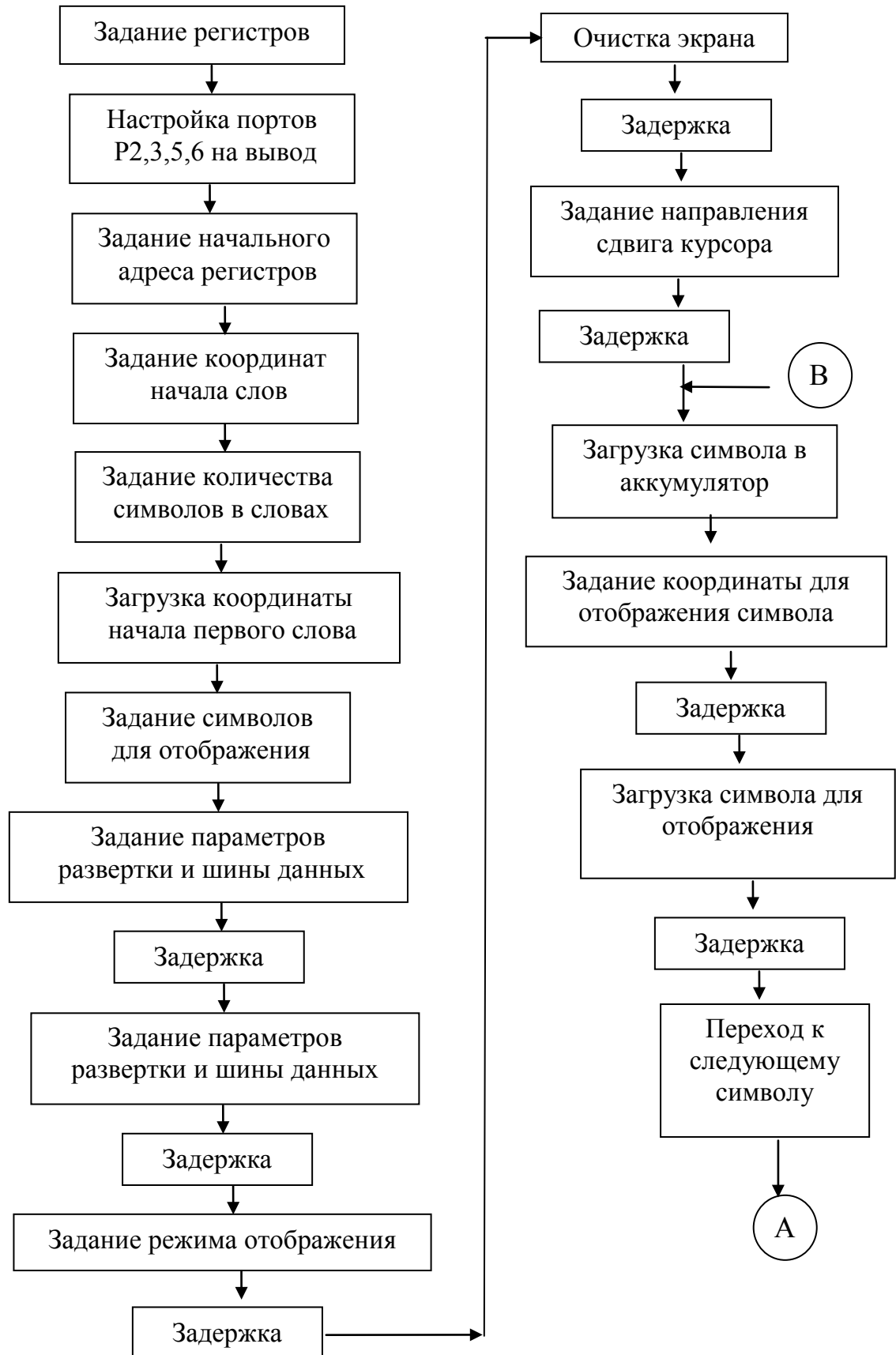


Рис.6.7.Схема подключения жидкокристаллического индикатора к лабораторному стенду

Алгоритм вывода данных на индикатор должен включать задание координат начала слов, задание количества символов в слове, выдачу символов, задание параметров развертки и шины данных рис 6.8.



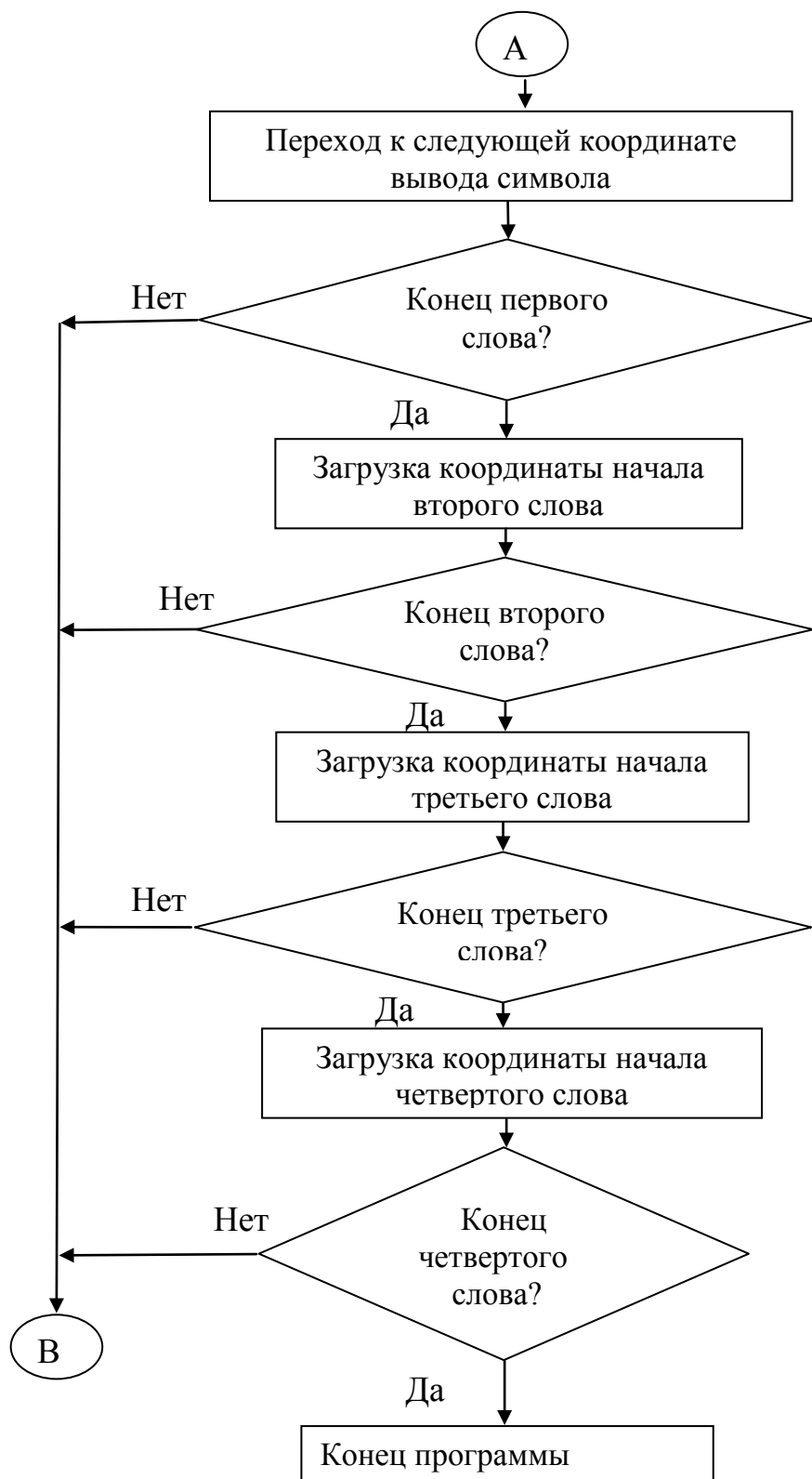


Рис.6.8. Алгоритм управления жидкокристаллическим индикатором

Пример программы для вывода слов в четыре строки
**«ТОМСКИЙ
ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ
КАФЕДРА ПМЭ»**

приведен ниже.

```
;-----
; Lab_Test.ASM программа учебно-лабораторного стенда.
; Команда $8F - программный сброс контроллера. Ответ: 70$, 'O', 'K', 'S', 'b', 'r', CRC8
; Команда $80 - Запрос версии ПО в контроллере. Ответ 7F$, 'I', '4', 'N', 'I', '0', CRC8
; Команда $81, D5, D7, D8, P0.7, P0.6, x, CRC8 - вывод данных в соответствующие регистры.
; Команда /$82, D5, D7, D8, P0.7, P0.6, x, CRC8 - Запрос значений в регистрах D5, D7, D8.
; Команда $83, x, x, x, x, N, CRC8 Запрос N измерения данных с ADC0 и ADC1.
; Команда вывести значение в DAC0 и DAC1 $84, DAC0H, DAC0L, DAC1H, DAC1L, x, CRC8
; Команда сделать N выборок через время th+tl в микросекундах, Tzh+Tzl.
; $85, N, th, tl, Tzh, Tzl, CRC8
```

```
;-----
$INCLUDE (C8051F060.inc)
```

```
;-----
;ЗАДАНИЕ РЕГИСТРОВ
```

```
R8 EQU 0X08;
R9 EQU 0X09;
R10 EQU 0X0A;
R11 EQU 0X0B;
R12 EQU 0X0C;
R13 EQU 0X0D;
R14 EQU 0X0E;
R15 EQU 0X0F;
R16 EQU 0X10;
R17 EQU 0X11;
R18 EQU 0X12;
R19 EQU 0X13;
R20 EQU 0X14;
R21 EQU 0X15;
R22 EQU 0X16;
R23 EQU 0X17;
R24 EQU 0X18;
R25 EQU 0X19;
R26 EQU 0X1A;
R27 EQU 0X1B;
R28 EQU 0X1C;
R29 EQU 0X1D;
R30 EQU 0X1E;
R31 EQU 0X1F;
R32 EQU 0X20;
R33 EQU 0X21;
R34 EQU 0X22;
R35 EQU 0X23;
R36 EQU 0X24;
R37 EQU 0X25;
R38 EQU 0X26;
R39 EQU 0X27;
R40 EQU 0X28;
R41 EQU 0X29;
```

```

R42 EQU 0X2A;
R43 EQU 0X2B;
R44 EQU 0X2C;
R45 EQU 0X2D;
R46 EQU 0X2E;
R47 EQU 0X2F;
R48 EQU 0X30;
R49 EQU 0X31;
R50 EQU 0X32;
R51 EQU 0X33;
R52 EQU 0X34;
R53 EQU 0X35;
R54 EQU 0X36;
R55 EQU 0X37;
R56 EQU 0X38;
R57 EQU 0X39;

;-----
; настройка стека
;-----

STACK SEGMENT IDATA ; выбор сегмента стека
RSEG STACK
DS 80h ; резервирование 128 байтов для стека
;-----
; настройка таймеров
;-----

CSEG AT 0
ljmp Main

Temp_2 SEGMENT CODE;
RSEG Temp_2 ; выбор сегмента CODE
USING 0 ; использование нулевого банка регистров

Main:
mov WDTCN, #0deh ; включение watchdog timer
mov WDTCN, #0adh
mov SP, #STACK-1 ; инициализация указателя стека

mov SFRPAGE, #0FH ; выбор страницы 0fH
MOV XBR2, #40H

MOV P3MDOUT, #0FFH ; настройка портов на вывод
MOV P2MDOUT, #0FFH ; настройка портов на вывод
MOV P5MDOUT, #0FFH ; настройка портов на вывод
MOV P6MDOUT, #0FFH ; настройка портов на вывод
;-----
MOV R0, #0EH ;
MOV R1, #0X ; ЗАДАНИЕ НАЧАЛЬНОГО АДРЕСА РЕГИСТРОВ
;-----
MOV R2, #089H ; КООРДИНАТА НАЧАЛА ПЕРВОГО СЛОВА
MOV R3, #0C5H ; КООРДИНАТА НАЧАЛА ВТОРОГО СЛОВА
MOV R4, #097H ; КООРДИНАТА НАЧАЛА ТРЕТЬЕГО СЛОВА
MOV R5, #0D7H ; КООРДИНАТА НАЧАЛА ЧЕТВЕРТОГО СЛОВА
;-----
MOV R6, #007H ; КОЛИЧЕСТВО СИМВОЛОВ В ПЕРВОМ СЛОВЕ
MOV R7, #010H ; КОЛИЧЕСТВО СИМВОЛОВ ВО ВТОРОМ СЛОВЕ

```

```

MOV R11,#00CH ; КОЛИЧЕСТВО СИМВОЛОВ В ТРЕТЬЕМ СЛОВЕ
MOV R13,#00CH ; КОЛИЧЕСТВО СИМВОЛОВ В ЧЕТВЕРТОМ СЛОВЕ
;-----
MOV R12,R2 ; ЗАГРУЗКА КООРДИНАТЫ НАЧАЛА ПЕРВОГО
СЛОВА
MOV P6,#00H ; СБРОС ПОРТА P6
;-----
;ЗАДАНИЕ СИМВОЛОВ ДЛЯ ОТОБРАЖЕНИЯ
;-----
;ЗАДАНИЕ ПЕРВОГО СЛОВА
MOV R14,#054H ;СИМВОЛ(Т)
MOV R15,#06fH ;СИМВОЛ(О)
MOV R16,#0BcH ;СИМВОЛ(М)
MOV R17,#063H ;СИМВОЛ(С)
MOV R18,#0baH ;СИМВОЛ(К)
MOV R19,#0b8H ;СИМВОЛ(И)
MOV R20,#0b9H ;СИМВОЛ(Й)
;ЗАДАНИЕ ВТОРОГО СЛОВА
MOV R21,#0a8H ;СИМВОЛ(П)
MOV R22,#06fH ;СИМВОЛ(О)
MOV R23,#0bbH ;СИМВОЛ(Л)
MOV R24,#0b8H ;СИМВОЛ(И)
MOV R25,#0BfH ;СИМВОЛ(Т)
MOV R26,#065H ;СИМВОЛ(Е)
MOV R27,#078H ;СИМВОЛ(Х)
MOV R28,#0bdH ;СИМВОЛ(Н)
MOV R29,#0b8H ;СИМВОЛ(И)
MOV R30,#0c0H ;СИМВОЛ(Ч)
MOV R31,#065H ;СИМВОЛ(Е)
MOV R32,#063H ;СИМВОЛ(С)
MOV R33,#0baH ;СИМВОЛ(К)
MOV R34,#0b8H ;СИМВОЛ(И)
MOV R35,#0b9H ;СИМВОЛ(Й)
;ЗАДАНИЕ ТРЕТЬЕГО СЛОВА
MOV R36,#0A9H ;СИМВОЛ(У)
MOV R37,#0BDH ;СИМВОЛ(Н)
MOV R38,#0B8H ;СИМВОЛ(И)
MOV R39,#0B3H ;СИМВОЛ(В)
MOV R40,#065H ;СИМВОЛ(Е)
MOV R41,#070H ;СИМВОЛ(Р)
MOV R42,#063H ;СИМВОЛ(С)
MOV R43,#0B8H ;СИМВОЛ(И)
MOV R44,#0BFH ;СИМВОЛ(Т)
MOV R45,#065H ;СИМВОЛ(Е)
MOV R46,#0BFH ;СИМВОЛ(Т)
;ЗАДАНИЕ ЧЕТВЕРТОГО СЛОВА
MOV R47,#04BH ;СИМВОЛ(К)
MOV R48,#061H ;СИМВОЛ(А)
MOV R49,#0E4H ;СИМВОЛ(Ф)
MOV R50,#065H ;СИМВОЛ(Е)
MOV R51,#0E3H ;СИМВОЛ(Д)
MOV R52,#070H ;СИМВОЛ(Р)
MOV R53,#061H ;СИМВОЛ(А)
MOV R54,#05FH ;пробел
MOV R55,#0A8H ;СИМВОЛ(П)
MOV R56,#04DH ;СИМВОЛ(М)
MOV R57,#0AFH ;СИМВОЛ(Э)

```

```

;-----
;основная программа
;-----
;ИНИЦИАЛИЗАЦИЯ ДИСПЛЕЯ
MOV P3,#11110001B
MOV P3,#11110000B ; ФОРМИРОВАНИЕ ИМПУЛЬСА ВЫБОРА РЕГИСТРА
ACALL ZADERGKA ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
SETB P3.2 ; НАЧАЛО СИНХРОНИЗАЦИИ
ACALL ZADERGKA ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
MOV P2,#00111111B ; ОПРЕДЕЛЕНИЕ ПАРАМЕТРОВ РАЗВЕРТКИ И
; ШИРИНЫ ШИНЫ ДАННЫХ
ACALL ZADERGKA ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
CLR P3.2 ; КОНЕЦ СИНХРОНИЗАЦИИ
ACALL ZADERGKA1 ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
;-----
MOV P3,#11110001B ;
MOV P3,#11110000B ; ФОРМИРОВАНИЕ ИМПУЛЬСА ВЫБОРА РЕГИСТРА
ACALL ZADERGKA ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
SETB P3.2 ; НАЧАЛО СИНХРОНИЗАЦИИ
ACALL ZADERGKA ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
MOV P2,#00111111B ; ОПРЕДЕЛЕНИЕ ПАРАМЕТРОВ РАЗВЕРТКИ И
; ШИРИНЫ ШИНЫ ДАННЫХ
ACALL ZADERGKA ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
CLR P3.2 ; КОНЕЦ СИНХРОНИЗАЦИИ
ACALL ZADERGKA1 ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
;-----
MOV P3,#11110001B ;
MOV P3,#11110000B ; ФОРМИРОВАНИЕ ИМПУЛЬСА ВЫБОРА РЕГИСТРА
ACALL ZADERGKA ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
SETB P3.2 ; НАЧАЛО СИНХРОНИЗАЦИИ
ACALL ZADERGKA ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
MOV P2,#00001100B ; ВЫБОР РЕЖИМА ОТОБРАЖЕНИЯ
ACALL ZADERGKA ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
CLR P3.2 ; КОНЕЦ СИНХРОНИЗАЦИИ
ACALL ZADERGKA1 ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
;-----
MOV P3,#11110001B ;
MOV P3,#11110000B ; ФОРМИРОВАНИЕ ИМПУЛЬСА ВЫБОРА РЕГИСТРА
ACALL ZADERGKA ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
SETB P3.2 ; НАЧАЛО СИНХРОНИЗАЦИИ
ACALL ZADERGKA ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
MOV P2,#00000001B ; ОЧИСТКА ЭКРАНА
ACALL ZADERGKA ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
CLR P3.2 ; КОНЕЦ СИНХРОНИЗАЦИИ
ACALL ZADERGKA1 ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
;-----
MOV P3,#11110001B ;
MOV P3,#11110000B ; ФОРМИРОВАНИЕ ИМПУЛЬСА ВЫБОРА РЕГИСТРА
ACALL ZADERGKA ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
SETB P3.2 ; НАЧАЛО СИНХРОНИЗАЦИИ
ACALL ZADERGKA ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
MOV P2,#00000111B ; ВЫБОР НАПРАВЛЕНИЯ СДВИГА КУРСОРА
ACALL ZADERGKA ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
CLR P3.2 ; КОНЕЦ СИНХРОНИЗАЦИИ
ACALL ZADERGKA1 ; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
;-----
;ВЫВОД СЛОВ НА ДИСПЛЕЙ

```


POVTR:	
MOV A,@R0	; ЗАГРУЗКА СИМВОЛА В АККУМУЛЯТОР
MOV P3,#11110001B	;
MOV P3,#11110000B	; ФОРМИРОВАНИЕ ИМПУЛЬСА ВЫБОРА РЕГИСТРА
ACALL ZADERGKA	; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
SETB P3.2	; НАЧАЛО СИНХРОНИЗАЦИИ
ACALL ZADERGKA	; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
MOV P2,R12	; ПРИСВОЕНИЕ СЧЕТЧИКУ АДРЕСА В ОБЛАСТИ DDRAM
ACALL ZADERGKA	; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
CLR P3.2	; КОНЕЦ СИНХРОНИЗАЦИИ
ACALL ZADERGKA1	; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
;-----	
MOV P3,#11110000B	;
MOV P3,#11110001B	; ФОРМИРОВАНИЕ ИМПУЛЬСА ВЫБОРА РЕГИСТРА
ACALL ZADERGKA	; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
SETB P3.2	; НАЧАЛО СИНХРОНИЗАЦИИ
ACALL ZADERGKA	; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
MOV P2,A	; ВЫБОР СИМВОЛА ДЛЯ ОТОБРАЖЕНИЯ
ACALL ZADERGKA	; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
CLR P3.2	; КОНЕЦ СИНХРОНИЗАЦИИ
ACALL ZADERGKA1	; ВЫЗОВ ПОДПРОГРАММЫ ЗАДЕРЖКИ
;-----	
INC R0	; ПЕРЕХОД К СЛЕДУЮЩЕМУ СИМВОЛУ
INC R12	; ПЕРЕХОД К СЛЕДУЮЩЕМУ АДРЕСУ В ОБЛАСТИ DDRAM
JB P6.0,SLOVO2	;
DJNZ R6,POVTR	; ПРОВЕРКА ДЛИНЫ ПЕРВОГО СЛОВА
;-----	
SLOVO2:	
JB P6.3,PEREHOD	;
MOV R12,R3	; ЗАГРУЗКА КООРДИНАТЫ НАЧАЛА ВТОРОГО СЛОВА
SETB P6.0	;
SETB P6.3	;
PEREHOD:	
JB P6.1,SLOVO3	;
DJNZ R7,POVTR	; ПРОВЕРКА ДЛИНЫ ВТОРОГО СЛОВА
;-----	
SLOVO3:	
JB P6.4,PEREHOD1	;
MOV R12,R4	; ЗАГРУЗКА КООРДИНАТЫ НАЧАЛА ТРЕТЬЕГО СЛОВА
SETB P6.1	;
SETB P6.4	;
PEREHOD1:	
JB P6.2,SLOVO4	;
DJNZ R11,POVTR	; ПРОВЕРКА ДЛИНЫ ТРЕТЬЕГО СЛОВА
;-----	
SLOVO4:	
JB P6.5,PEREHOD2	;
MOV R12,R5	; ЗАГРУЗКА КООРДИНАТЫ НАЧАЛА ЧЕТВЕРТОГО СЛОВА
SETB P6.2	;
SETB P6.5	;
PEREHOD2:	
DJNZ R13,POVTR	; ПРОВЕРКА ДЛИНЫ ЧЕТВЕРТОГО СЛОВА

```

;-----
REPIAT:      JMP    REPIAT      ; КОНЕЦ ПРОГРАММЫ
;-----
;ПОДПРОГРАММЫ ЗАДЕРЖКИ
ZADERGKA:
MOV  R10,#0FFH      ;
G10:  DJNZ  R10,G10    ; ЗАДЕРЖКА
      RET              ;
;-----
ZADERGKA1:
MOV  R10,#0FFH      ;
G11:  DJNZ  R10,G11    ;
MOV  R10,#0FFH      ; ЗАДЕРЖКА
G12:  DJNZ  R10,G12    ;
      RET              ;
;-----
;
      END

```

ПРОГРАММА РАБОТЫ

- 1.Подключить лабораторный стенд к персональному компьютеру через разъемы для внешних устройств.
- 2.Подключить индикаторный ЖК- модуль к лабораторному стенду через 20 контактный разъем **X**.
- 3.Составить алгоритм вывода информации, заданной преподавателем, на индикатор.
- 4.Составить программу вывода информации на индикаторы.
- 6.Вывести записанную информацию на информационное табло.
- 5.Изменяя напряжение V_0 снять зависимость изменения контраста от напряжения.
- 7.Определить максимальную частоту смены информации.
- 8.Определить пороговое напряжение индикатора.

СОДЕРЖАНИЕ ОТЧЕТА

- 1.Структурную схему блока индикации.
- 2.Принципиальную схему блока управления модуля жидкокристаллического индикатора.
- 3.Алгоритм управления модулем жидкокристаллического индикатора.
4. Программу для вывода информации на модуль жидкокристаллического индикатора.
- 5.Таблицу знакогенератора используемых символов.
- 6.Зависимость контраста от управляющего напряжения.
7. Расчет порогового напряжения индикатора
- 10.Выводы.

11. Ответы на контрольные вопросы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. К каким типам индикаторов относится жидкокристаллический индикатор?
2. Как определяется контрастность изображения?
3. Почему для управления жидкокристаллическим индикатором применяется переменное напряжение?
4. Назовите преимущества жидкокристаллических индикаторов по сравнению с полупроводниковыми.
5. Какие свойства жидкокристаллического вещества используют при производстве жидкокристаллических индикаторов?
6. Дайте определение времени включения и времени выключения индикатора.
7. Какой метод индикации применен в лабораторной работе, какие еще методы существуют?
8. Нарисуйте принципиальную схему управления матричным жидкокристаллическим индикатором.
9. Перечислите основные параметры и характеристики матричных жидкокристаллических индикаторов.
10. Как можно регулировать контрастность изображения на матричном жидкокристаллическом индикаторе?
11. Почему жидкокристаллические индикаторы относятся к пассивным типам индикаторов?
12. Чем отличаются пассивная и активная матрица в жидкокристаллических дисплеях?

ОПИСАНИЕ СРЕДСТВ ПРОГРАММИРОВАНИЯ И ОТЛАДКИ МИКРОКОНТРОЛЛЕРА C8051

Микроконтроллеры фирмы Silicon Laboratories C8051Fxxx оптимально подходят для построения устройств, требующих высокой производительности, точности, большой степени интеграции и малого потребления. Они программно совместимы с 8051-м стандартом, но одновременно имеют рекордно высокую производительность – до 100 MIPS. Микроконтроллеры Silabs также уникальны своей интеграцией с аналоговыми узлами. Многие модели имеют до двух встроенных независимых АЦП, встроенный ЦАП, компараторы напряжения, программируемые усилители напряжения и др.

Краткие сведения о Cygnal C8051F060

Микроконтроллеры (МК) C8051F06x представляют собой полностью интегрированные на одном кристалле системы для обработки смешанных сигналов (аналога-цифровых), которые имеют 59 (C8051F060/2/4/6) или 24 (C8051F061/3/5/7) цифровых входа/выхода, а также 2 встроенных 16-разрядных АЦП с производительностью 1 MIPS. Отличительные особенности данного микроконтроллера:

- Высокопроизводительное микропроцессорное ядро CIP-51 с конвейерной архитектурой, совместимое со стандартом 8051 (максимальная производительность – 25 MIPS).
- Встроенные 16-разрядные АЦП (производительность - 1 млн. преобразований в секунду) с контроллером прямого доступа к памяти.
- Контроллер локальной сети (CAN 2.0B) с 32 объектами сообщений, каждое из которых имеет собственную маску идентификатора (C8051F060/1/2/3).
- Встроенные средства отладки, обеспечивающие внутрисистемную, «неразрушающую» отладку в режиме реального времени.
- 10-разрядный 8-канальный АЦП (максимальная производительность – 200 тыс. преобр./сек.) с программируемым усилителем и аналоговым мультиплексором (C8051F060/1/2/3).
- Два 12-разрядных цифро-аналоговых преобразователя с программируемым обновлением выходного сигнала (C8051F060/1/2/3).
- 64 Кб (C8051F060/1/2/3/4/5) или 32 Кб (C8051F066/7) Flash-памяти, программируемой внутрисистемно.
- 4352 (4096+256) байт встроенного ОЗУ.

- Интерфейс внешней памяти данных с доступным адресным пространством 64 Кб (C8051F060/2/4/6).
- I2C/SMBus, SPI и два УАПП аппаратно реализованные последовательные интерфейсы.
- 5шт. 16-разрядных таймеров общего назначения.
- 6 модулей захвата/сравнения программируемые через массив счетчиков/таймеров (ПМС).
- Встроенные сторожевой таймер, схема слежения за напряжением и температурой.

Архитектура микроконтроллера C8051F060 представлена на рисунке А.1.

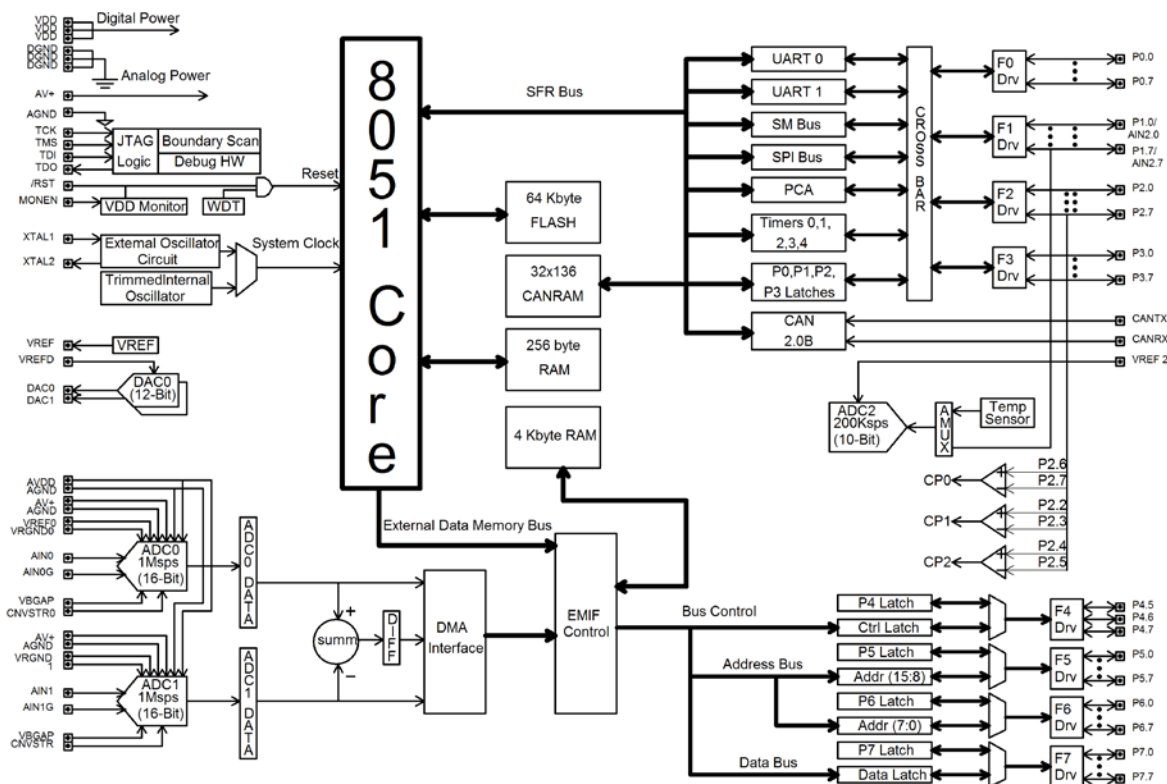


Рис.А.1. Архитектура микроконтроллера C8051F060

Память программ и стек

CIP-51 имеет адресное пространство памяти программ 64 Кбайт. В МК C8051F060 физически реализовано 64 Кбайт этой памяти программ, которая является внутрисистемной перепрограммируемой Flash-памятью, занимающей непрерывный блок адресов от 0x0000 до 0xFFFF рисунок А.2.

Следует иметь ввиду, что 1024 байт (0xFC00 – 0xFFFF) этой памяти зарезервированы для целей производителя и не доступны для хранения программ пользователя.

По умолчанию память программ настраивается только для чтения. Однако CIP-51 может записывать данные в память программ, для чего необходимо разрешить запись во Flash-память программ. Эта возможность позволяет CIP-51 обновлять программный код и использовать память программ для долговременного хранения данных.

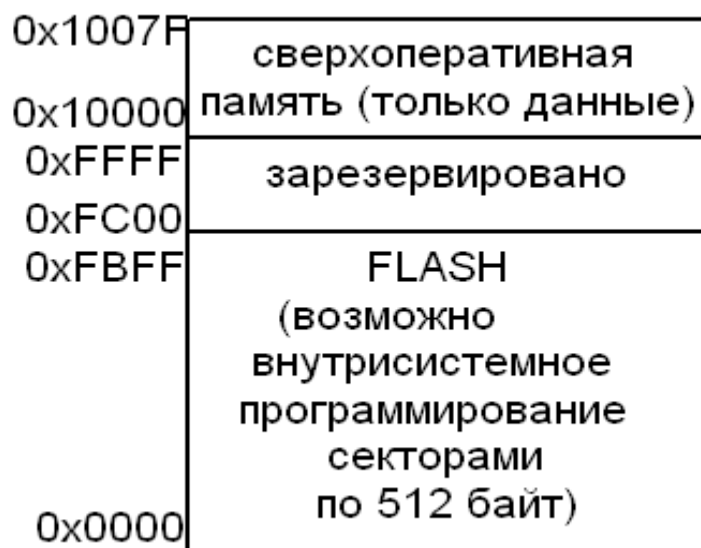


Рис.А.2. Память программ/данных (Flash)

Программный стек может быть размещен в любом месте 256-байтной памяти данных. Область стека определяется с использованием указателя стека (Stack Pointer – SP, 0x81). SP будет указывать на последнюю использованную ячейку. Следующее значение, загружаемое в стек, размещается по адресу SP+1, и затем SP инкрементируется. При сбросе SP инициализируется значением 0x07. Поэтому первое значение, загружаемое в стек, размещается по адресу 0x08, которое также является первым регистром (R0) регистрового банка 1. Таким образом, если требуется использовать более одного банка регистров, SP следует инициализировать адресом ячейки ОЗУ, не используемой для хранения данных. Стек может иметь глубину до 256 байт.

МК также имеют встроенный аппаратный регистратор стековых операций, который представляет собой 32-разрядный сдвиговый регистр. Каждая команда PUSH или инкремент SP загружают один регистрационный бит в этот регистр, каждая команда CALL или прерывание загружают два регистрационных бита в этот регистр. Схема

регистратора стековых операций способна определять переполнение или опустошение стека и может уведомлять программные средства отладки, даже если МК отлаживается в режиме реального времени.

Память данных

Физически реализовано 256 байт внутреннего ОЗУ, отображенного в пространстве памяти данных с адресами от 0x00 до 0xFF. Младшие 128 байт памяти данных используются для регистров общего назначения (РОН) и сверхоперативного ЗУ (СОЗУ). Для доступа к младшим 128 байтам памяти данных можно использовать либо прямую, либо косвенную адресацию. Ячейки с адресами от 0x00 до 0x1F разбиты на четыре банка РОН, каждый банк состоит из восьми однобайтовых регистров. Следующие 16 байт (0x20 – 0x2F) могут адресоваться побайтно или побитно как 128 бит, доступные в режиме прямой битовой адресации.

Старшие 128 байт памяти данных доступны только в режиме косвенной адресации. Эта область памяти занимает то же самое адресное пространство, что и регистры специального назначения (Special Function Registers - SFR), но физически отделена от них. При обращении к ячейкам памяти с адресами 0x7F – 0xFF использующийся в команде режим адресации определяет, к чему осуществляется доступ: к старшим 128 байтам памяти данных или к SFR. Команды, которые используют режим прямой адресации, будут обращаться к SFR. Команды, использующие режим косвенной адресации, будут обращаться к старшим 128 байтам памяти данных.

Страничная организация SFR

В CIP-51 используется страничная организация SFR, что позволяет отображать в адресном пространстве 0x80 – 0xFF большое количество регистров SFR. Пространство памяти SFR имеет 256 страниц. Таким образом, каждая ячейка памяти из области 0x80 – 0xFF может адресовать до 256 регистров SFR. В МК семейства C8051F060 используется пять SFR страниц: 0,1,2,3 и F. SFR страницы выбираются при помощи регистра выбора страницы SFR SFRPAGE. Последовательность действий при чтении и записи SFR следующая:

1. Выбрать номер соответствующей SFR страницы, используя регистр SFRPAGE.
2. Прочитать или записать регистр SFR, используя режим прямой адресации.

Прерывания

При возникновении прерывания регистр страницы SFR автоматически переключится на ту страницу SFR, которая содержит флаг, вызвавший прерывание. Функция автоматического переключения SFR страницы просто освобождает процедуру обслуживания прерывания от необходимости переключать SFR страницы. После выполнения команды RETI SFR страница автоматически восстановится, то есть будет установлена та SFR страница, которая использовалась до прерывания. Старшим байтом стека является SFRPAGE, текущая SFR страница. Вторым байтом стека SFR страниц является SFRNEXT. Третьим (младшим) байтом стека SFR страниц является SFRLAST рисунок А.3. При возникновении прерывания текущее значение SFRPAGE загружается в регистр SFRNEXT, а значение SFRNEXT загружается в регистр SFRLAST. Затем в регистр SFRPAGE аппаратно записывается номер той SFR страницы, которая содержит флаг, вызвавший прерывание. При возврате из прерывания стек SFR страниц «выталкивается», в результате чего значение из SFRNEXT возвращается в регистр SFRPAGE, то есть номер SFR страницы восстанавливается без вмешательства со стороны программы. Значение из SFRLAST помещается в регистр SFRNEXT. При необходимости в процедуре обслуживания прерывания можно модифицировать значения, сохраненные в регистрах SFRPAGE и SFRLAST, тогда при выполнении команды RETI произойдет возврат на другую SFR страницу. Модификация регистров в стеке SFR страниц не приведет к «проталкиванию» стека. Лишь вызов прерывания и возврат из прерывания могут инициировать операции загрузки/извлечения стека SFR страниц.

Автоматическое аппаратное переключение страниц SFR можно разрешить или запретить, используя бит разрешения автоматического управления страницами SFR, размещенный в регистре управления страницами SFR. После сброса эта функция по умолчанию разрешена. Таким образом, функция автоматического переключения SFR страниц будет включена до тех пор, пока не будет отключена в программе.

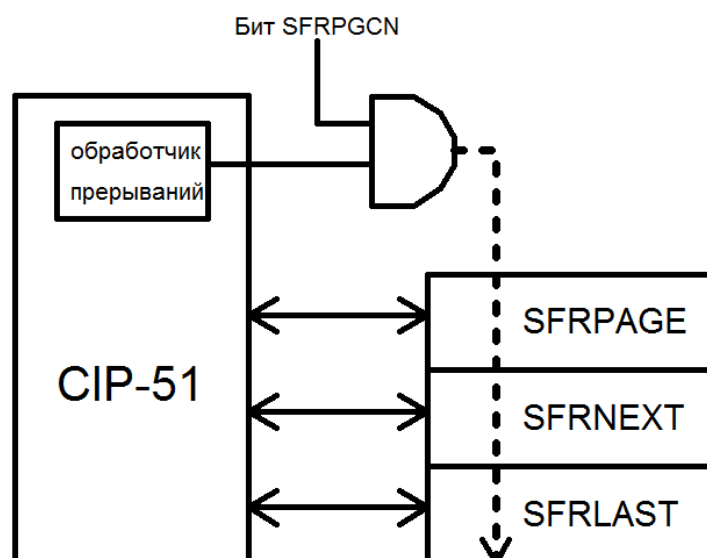


Рис.А.3. Стек страниц SFRPAGE

Описание регистров SFR

Регистры SFR используются для разрешения/запрещения источников прерываний и установки их приоритетов. В таблице А.1 представлен регистр разрешения прерываний IE, в таблице А.2 дополнительный регистр разрешения прерываний 1 EIE1, в таблице А.3 дополнительный регистр разрешения прерываний 2 EIE2.

IE: Регистр разрешения прерываний

Таблица А.1

№ бита	7	6	5	4	3	2	1	0
Имя бита	EA	IEGF0	ET2	ES0	ET1	EX1	ET0	EX0
Доступность	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

SFR Адрес: 0xA8

SFR страница: Все

Бит 7: EA: Бит разрешения всех прерываний.

Это бит глобально разрешает/запрещает все прерывания. Будучи сброшенным в 0, он перекрывает индивидуальные маски прерываний :

- 0: Все источники прерываний запрещены.
- 1: Каждое прерывание разрешено/запрещено в соответствии с его индивидуальной маской.

Бит 6: IEGF0: Флаг общего назначения 0.

Это флаг общего назначения, предназначенный для использования под управлением программы.

Бит 5: ET2: Бит разрешения прерывания от Таймера 2.

Этот бит устанавливает маскирование прерывания от Таймера 2 :

- 0: Все прерывания от Таймера 2 запрещены.
- 1: Разрешены запросы прерываний, генерируемые при установке флага TF2 (T2CON.7)

Бит 4: ES0: Бит разрешения прерываний от последовательного порта УАПП0.

Этот бит устанавливает маскирование прерывания от последовательного порта УАПП0 :

- 0: Прерывания от УАПП0 запрещены.
- 1: Прерывания от УАПП0 разрешены.

Бит 3: ET1: Бит разрешения прерывания от Таймера 1.

Этот бит устанавливает маскирование прерывания от Таймера 1 :

- 0: Все прерывания от Таймера 1 запрещены.
- 1: Разрешены запросы прерываний, генерируемые при установке флага TF1 (TCON.7).

Бит 2: EX1: Бит разрешения внешнего прерывания 1.

Этот бит устанавливает маскирование внешнего прерывания 1 :

- 0: Внешнее прерывание 1 запрещено.
- 1: Разрешены запросы прерываний, генерируемые сигналом на выводе /INT1.

Бит 1: ET0: Бит разрешения прерывания от Таймера 0.

Этот бит устанавливает маскирование прерывания от Таймера 0 :

- 0: Все прерывания от Таймера 0 запрещены.
- 1: Разрешены запросы прерываний, генерируемые при установке флага TF0 (TCON.5).

Бит 0: EX0: Бит разрешения внешнего прерывания 0.

Этот бит устанавливает маскирование внешнего прерывания 0 :

- 0: Внешнее прерывание 0 запрещено.
- 1: Разрешены запросы прерываний, генерируемые сигналом на выводе /INT0.

EIE1: Дополнительный регистр разрешения прерываний 1

Таблица A.2

№ бита	7	6	5	4	3	2	1	0
Имя бита	EAD C0	CP2I E	CP1I E	CP0I E	EPCA 0	EWAD C0	ESM B0	ESPI 0
Доступность	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

SFR Адрес: 0xE6

SFR страница: Все

Бит 7: EADC0: Бит разрешения прерываний, возникающих при завершении преобразования АЦП0.

Этот бит устанавливает маскирование прерывания, возникающего при завершении преобразования АЦП0:

- 0: Прерывание, возникающее при завершении преобразования АЦП0, запрещено.
- 1: 0: Прерывание, возникающее при завершении преобразования АЦП0, разрешено

Бит 6: CP2IE: Бит разрешения прерываний от Компаратора 2 (CP2).

Этот бит устанавливает маскирование прерывания от CP2 :

- 0: Прерывания от CP2 запрещены.
- 1: Разрешены запросы прерываний, генерируемые при установке флага CP2IF.

Бит 5.

Этот бит устанавливает маскирование прерывания от CP1 :

- 0: Прерывания от CP1 запрещены.
- 1: Разрешены запросы прерываний, генерируемые при установке флага CP1IF.

Бит 4: CP0IE: Бит разрешения прерываний от Компаратора 0 (CP0).

Этот бит устанавливает маскирование прерывания от CP0 :

- 0: Прерывания от CP0 запрещены.
- 1: Разрешены запросы прерываний, генерируемые при установке флага CP0IF.

Бит 3: ERCA0: Бит разрешения прерываний от программируемого массива счетчиков (PCA0).

Этот бит устанавливает маскирование прерывания от PCA0 :

- 0: Все прерывания от PCA0 запрещены.
- 1: Разрешены запросы прерываний, генерируемые PCA0.

Бит 2: EWADC0: Бит разрешения прерываний от детектора диапазона АЦП0.

Этот бит устанавливает маскирование прерывания от детектора диапазона АЦП0 :

- 0: Прерывание от детектора диапазона АЦП0 запрещено.
- 1: Разрешены запросы прерываний, генерируемые детектором диапазона АЦП0.

Бит 1: ESMB0: Бит разрешения прерываний от модуля SMBus.

Этот бит устанавливает маскирование прерывания от модуля SMBus :

- 1: Разрешены запросы прерываний, генерируемые при установке флага SI.

Бит 0: ESPI0: Бит разрешения прерываний от модуля SPI0.

Этот бит устанавливает маскирование прерывания от модуля SPI0:

- 0: Все прерывания от модуля SPI0 запрещены.
- 1: Разрешены запросы прерываний, генерируемые при установке флага SPI0.

EIE2: Дополнительный регистр разрешения прерываний 2

Таблица А.3

№ бита	7	6	5	4	3	2	1	0
Имя бита	EDMA 0	ES 1	ECAN 0	EADC 2	EWADC 2	ET 4	EADC 1	ET 3
Доступность	R/W	R/ W	R/W	R/W	R/W	R/ W	R/W	R/ W

SFR Адрес: 0Хе7

SFR страница: Все

Бит 7: EDMA0: Бит разрешения прерываний от интерфейса прямого доступа к памяти DMA0.

Этот бит устанавливает маскирование прерываний от модуля DMA0:

- 0: Прерывания от модуля DMA0 запрещены.
- 1: Прерывания от модуля DMA0 разрешены.

Бит 6: ES1: Бит разрешения прерываний от последовательного порта УАПП1.

Этот бит устанавливает маскирование прерывания от последовательного порта УАПП1 :

- 0: Прерывания от УАПП1 запрещены.
- 1: Прерывания от УАПП1 разрешены.

Бит 5: ECAN0: Бит разрешения прерываний от контроллера CAN.

Этот бит устанавливает маскирование прерываний от контроллера CAN :

- 0: Прерывание от контроллера CAN запрещено.
- 1: Разрешены запросы прерываний, генерируемые контроллером CAN.

Бит 4: EADC2: Бит разрешения прерываний, возникающих при завершении преобразования АЦП2.

Этот бит устанавливает маскирование прерывания, возникающего при завершении преобразования АЦП2 :

- 0: Прерывание, возникающее при завершении преобразования АЦП2, запрещено.
- 1: Разрешены запросы прерываний, генерируемые при завершении преобразования АЦП2.

Бит 3: EWADC2: Бит разрешения прерываний от детектора диапазона АЦП2.

Этот бит устанавливает маскирование прерываний от детектора диапазона АЦП2 :

- 0: Прерывание от детектора диапазона АЦП2 запрещено.
- 1: Разрешены запросы прерываний, генерируемые детектором диапазона АЦП2.

Бит 2: ET4: Бит разрешения прерываний от Таймера 4.

Этот бит устанавливает маскирование прерывания от Таймера 4 :

- 0: Все прерывания от Таймера 4 запрещены.
- 1: Разрешены запросы прерываний, генерируемые при установке флага TF4.

Бит 1: EADC1: Бит разрешения прерываний, возникающих при завершении преобразования АЦП1.

Этот бит устанавливает маскирование прерывания, возникающего при завершении преобразования АЦП1 :

- 0: Прерывание, возникающее при завершении преобразования АЦП1, запрещено.
- 1: Разрешены запросы прерываний, генерируемые при завершении преобразования АЦП1.

Бит 0: ET3: Бит разрешения прерываний от Таймера 3.

Этот бит устанавливает маскирование прерывания от Таймера 3 :

- 0: Все прерывания от Таймера 3 запрещены.
- 1: Разрешены запросы прерываний, генерируемые при установке флага TF3.

Обработка прерываний

CIP-51 имеет развитую систему прерываний, поддерживающую в общей сложности 22 источника прерываний с двумя уровнями приоритета. Распределение источников прерываний между встроенными периферийными модулями и внешними входными выводами зависит от конкретного типа МК. Каждый источник прерываний имеет один или несколько связанных с ним флагов прерываний, размещенных в SFR. Когда периферийный модуль или внешний источник прерываний регистрирует событие,

удовлетворяющее условию прерывания, соответствующий флаг прерывания устанавливается в 1.

Если прерывание от источника прерываний разрешено, то при установке флага прерывания генерируется запрос прерывания. Как только выполнение текущей команды завершится, будет сгенерирована команда LCALL перехода по предопределенному адресу, откуда начнется исполнение процедуры обслуживания прерывания. Каждая процедура обслуживания прерывания должна заканчиваться командой RETI, которая возвращает управление прерванной программе и приводит к выполнению той команды, которая исполнилась бы, если бы запроса прерывания не было. Если прерывания не разрешены, флаг прерывания игнорируется и выполнение программы продолжается в нормальном режиме.

Прерывание от каждого источника прерываний может быть разрешено или запрещено с помощью соответствующих битов разрешения прерываний в регистрах SFR(IE-EIE2). Однако, сначала прерывания необходимо разрешить глобально установкой в 1 бита EA (IE.7), только после этого состояние индивидуальных флагов разрешения прерываний будет иметь силу. Сброс в 0 бита EA запрещает прерывания от всех источников прерываний независимо от состояния индивидуальных флагов разрешения прерываний.

Некоторые флаги прерываний сбрасываются автоматически аппаратными средствами при переходе к процедуре ISR. Однако большинство флагов прерываний не сбрасываются аппаратно и должны быть сброшены программно до возвращения из процедуры ISR. Если флаг прерывания остается установленным после завершения выполнения команды возврата из прерывания, то сразу же будет сгенерирован новый запрос прерывания и после завершения выполнения следующей команды произойдет повторный переход к процедуре ISR.

Система команд микроконтроллера C8051F060 приведена в таблице А.4

Система команд

Таблица А.4

Мнемоника команды	описание	байты	циклы
	Арифметические операции		
ADD A,Rn	Сложение аккумулятора с регистром (n = 0...7)	1	1
ADD A,direct	Сложение аккумулятора с прямо-адресуемым байтом	2	2

ADD A,@Ri	Сложение аккумулятора с косвенно-адресуемым байтом ОЗУ	1	2
ADD A,#data	Сложение аккумулятора с константой	2	2
ADDC A,Rn	Сложение аккумулятора с регистром и переносом	1	1
ADDC A,direct	Сложение аккумулятора с прямо-адресуемым байтом и переносом	2	2
ADDC A,@Ri	Сложение аккумулятора с косвенно-адресуемым байтом ОЗУ и переносом	1	2
ADDC A,#data	Сложение аккумулятора с константой и переносом	2	2
SUBB A,Rn	Вычитание из аккумулятора регистра и заема	1	1
SUBB A,direct	Вычитание из аккумулятора прямо-адресуемого байта и заема	2	2
SUBB A,@Ri	Вычитание из аккумулятора косвенно-адресуемого байта ОЗУ и заема	1	2
SUBB A,#data	Вычитание из аккумулятора константы и заема	2	2
INC A	Инкремент аккумулятора	1	1
INC Rn	Инкремент регистра	1	1
INC direct	Инкремент прямо-адресуемого байта	2	2
INC @Ri	Инкремент косвенно-адресуемого байта ОЗУ	1	2
DEC A	Декремент аккумулятора	1	1
DEC Rn	Декремент регистра	1	1
DEC direct	Декремент прямо-адресуемого байта	2	2
DEC @Ri	Декремент косвенно-адресуемого байта ОЗУ	1	2
INC DPTR	Инкремент указателя данных	1	1
MUL AB	Умножение аккумулятора на регистр В	1	4
DIV AB	Деление аккумулятора на регистр В	1	8
DA A	Десятичная коррекция аккумулятора	1	1
	Логические операции		
ANL A,Rn	Логическое И аккумулятора и регистра	1	1
ANL A,direct	Логическое И аккумулятора и прямо-	2	2

	адресуемого байта		
ANL A,@Ri	Логическое И аккумулятора и косвенно-адресуемого байта ОЗУ	1	2
ANL A,#data	Логическое И аккумулятора и константы	2	2
ANL direct,A	Логическое И прямо-адресуемого байта и аккумулятора	2	2
ANL direct,#data	Логическое И прямо-адресуемого байта и константы	3	3
ORL A,Rn	Логическое ИЛИ аккумулятора и регистра	1	1
ORL A,direct	Логическое ИЛИ аккумулятора и прямо-адресуемого байта	2	2
ORL A,@Ri	Логическое ИЛИ аккумулятора и косвенно-адресуемого байта ОЗУ	1	2
ORL A,#data	Логическое ИЛИ аккумулятора и константы	2	2
ORL direct,A	Логическое ИЛИ прямо-адресуемого байта и аккумулятора	2	2
ORL direct,#data	Логическое ИЛИ прямо-адресуемого байта и константы	3	3
XRL A,Rn	Исключающее ИЛИ аккумулятора и регистра	1	1
XRL A,direct	Исключающее ИЛИ аккумулятора и прямо-адресуемого байта	2	2
XRL A,@Ri	Исключающее ИЛИ аккумулятора и косвенно-адресуемого байта ОЗУ	1	2
XRL A,#data	Исключающее ИЛИ аккумулятора и константы	2	2
XRL direct,A	Исключающее ИЛИ прямо-адресуемого байта и аккумулятора	2	2
XRL direct,#data	Исключающее ИЛИ прямо-адресуемого байта и константы	3	3
CLR A	Сброс аккумулятора	1	1
CPL A	Инверсия аккумулятора	1	1
RL A	Сдвиг аккумулятора влево циклический	1	1
RLC A	Сдвиг аккумулятора влево через перенос	1	1

RR A	Сдвиг аккумулятора вправо циклический	1	1
RRC A	Сдвиг аккумулятора вправо через перенос	1	1
SWAP A	Обмен местами тетрад в аккумуляторе	1	1
	Команды передачи данных		
MOV A,Rn	Пересылка в аккумулятор из регистра (n = 0...7)	1	1
MOV A,direct	Пересылка в аккумулятор прямо-адресуемого байта	2	2
MOV A,@Ri	Пересылка в аккумулятор косвенно-адресуемого байта ОЗУ	1	2
MOV A,#data	Загрузка в аккумулятор константы	2	2
MOV Rn,A	Пересылка в регистр из аккумулятора	1	1
MOV Rn,direct	Пересылка в регистр прямо-адресуемого байта	2	2
MOV Rn,#data	Загрузка в регистр константы	2	2
MOV direct,A	Пересылка по прямому адресу аккумулятора	2	2
MOV direct,Rn	Пересылка по прямому адресу регистра	2	2
MOV direct,direct	Пересылка прямо-адресуемого байта по прямому адресу	3	3
MOV direct,@Ri	Пересылка косвенно-адресуемого байта ОЗУ по прямому адресу	2	2
MOV direct,#data	Пересылка по прямому адресу константы	3	3
MOV @Ri,A	Пересылка в косвенно-адресуемую ячейку ОЗУ аккумулятора	1	2
MOV @Ri,direct	Пересылка в косвенно-адресуемую ячейку ОЗУ прямо-адресуемого байта	2	2
MOV @Ri,#data	Пересылка в косвенно-адресуемую ячейку ОЗУ константы	2	2
MOV DPTR,#data16	Загрузка указателя данных	3	3
MOVC A,@A+DPTR	Пересылка в аккумулятор байта из памяти программ	1	3

MOVC A,@A+PC	Пересылка в аккумулятор байта из памяти программ	1	3
MOVX A,@Ri	Пересылка в аккумулятор байта из внешней памяти данных	1	3
MOVX @Ri,A	Пересылка байта из аккумулятора во внешнюю память данных	1	3
MOVX A,@DPTR	Пересылка в аккумулятор из расширенной внешней памяти данных	1	3
MOVX @DPTR,A	Пересылка из аккумулятора в расширенную внешнюю память данных	1	3
PUSH direct	Загрузка в стек	2	2
POP direct	Извлечение из стека	2	2
XCH A,Rn	Обмен аккумулятора с регистром	1	1
XCH A,direct	Обмен аккумулятора с прямо-адресуемым байтом	2	2
XCH A,@Ri	Обмен аккумулятора с косвенно-адресуемым байтом ОЗУ	1	2
XCHD A,@Ri	Обмен младшей тетрады аккумулятора с младшей тетрадой косвенно-адресуемого байта ОЗУ	1	2
	Операции с битами		
CLR C	Сброс переноса	1	1
CLR bit	Сброс бита	2	2
SETB C	Установка переноса	1	1
SETB bit	Установка бита	2	2
CPL C	Инверсия переноса	1	1
CPL bit	Инверсия бита	2	2
ANL C,bit	Логическое И бита и переноса	2	2
ANL C,/bit	Логическое И инверсии бита и переноса	2	2
ORL C,bit	Логическое ИЛИ бита и переноса	2	2
ORL C,/bit	Логическое ИЛИ инверсии бита и переноса	2	2
MOV C,bit	Пересылка бита в перенос	2	2
MOV bit,C	Пересылка переноса в бит	2	2
JC rel	Переход, если перенос равен единице	2	2/3
JNC rel	Переход, если перенос равен нулю	2	2/3

JB bit,rel	Переход, если бит равен единице	3	3/4
JNB bit,rel	Переход, если бит равен нулю	3	3/4
JBC bit,rel	Переход, если бит установлен, с последующим сбросом бита	3	3/4
	Программные переходы		
ACALL addr11	Абсолютный вызов подпрограммы в пределах страницы в 2 Кбайта	2	3
LCALL addr16	Длинный вызов подпрограммы	3	4
RET	Возврат из подпрограммы	1	5
RETI	Возврат из подпрограммы обработки прерывания	1	5
AJMP addr11	Абсолютный переход внутри страницы в 2 Кбайта	2	3
LJMP addr16	Длинный переход в полном объеме памяти программ	3	4
SJMP rel	Короткий относительный переход внутри страницы в 256 байт	2	3
JMP @A+DPTR	Косвенный относительный переход	1	3
JZ rel	Переход, если аккумулятор равен нулю	2	2/3
JNZ rel	Переход, если аккумулятор не равен нулю	2	2/3
CJNE A,direct,rel	Сравнение аккумулятора с прямо-адресуемым байтом и переход, если не равно	3	3/4
CJNE A,#data,rel	Сравнение аккумулятора с константой и переход, если не равно	3	3/4
CJNE Rn,#data,rel	Сравнение регистра с константой и переход, если не равно	3	3/4
CJNE @Ri,#data,rel	Сравнение косвенно-адресуемого байта ОЗУ с константой и переход, если не равно	3	4/5
DJNZ Rn,rel	Декремент регистра и переход, если не нуль	2	2/3
DJNZ direct,rel	Декремент прямо-адресуемого байта и переход, если не нуль	3	3/4
NOP	Холостая команда	1	1

Описание отладочных средств

Существуют множество инструментальных средств, предназначенных для обеспечения и интенсификации труда разработчиков. Основное назначение любого отладчика – помочь разработчику разобраться на программно-аппаратном уровне в процессах, происходящих в устройстве, а затем добиться желаемых характеристик функционирования. Кроме этого инструментальные средства могут обладать дополнительными возможностями, которые избавляют разработчика от множества утомительных процедур. В данном руководстве рассмотрено применение внутрисхемного отладчика и интегрированная среда разработки Silicon Laboratories.

Общее описание внутрисхемного отладчика и Silicon Laboratories IDE

Silicon Laboratories IDE – интегрированная среда разработки, представляет собой программный продукт, работающий под управлением операционной системы Windows и предназначенный для написания, отладки и оптимизации программ для контроллеров Cygnal. Silicon Labs IDE включает в себя:

Project manager – менеджер проекта, который позволяет создавать и сохранять текущие проекты;

Editor – текстовый редактор для написания программы;

Simulator – симулятор для проверки функционирования программы;

Keil assembler – язык программирования;

Linker – утилита объединения всех файлов проекта;

Compiler - утилита преобразования программы в машинные коды.

ICD

ICD работает под управлением программной среды Silicon Laboratories и обладает следующими возможностями:

- Отладка в режиме реального времени с возможностью пошаговой отладки;
- Связь с компьютером по USB – интерфейсу;
- Просмотр и модификация содержимого управляющих регистров, RAM и EEPROM;
- Внутрисхемная отладка и встроенная система программирования контроллера Cygnal;
- Содержит обширную систему визуального вывода информации.

На рисунке А.5 представлена фотография лабораторного стенда, содержащая точечные, семисегментные и матричные индикаторы,

управляемые микроконтроллером C8051F060 работающего в программной среде Silicon Laboratories.

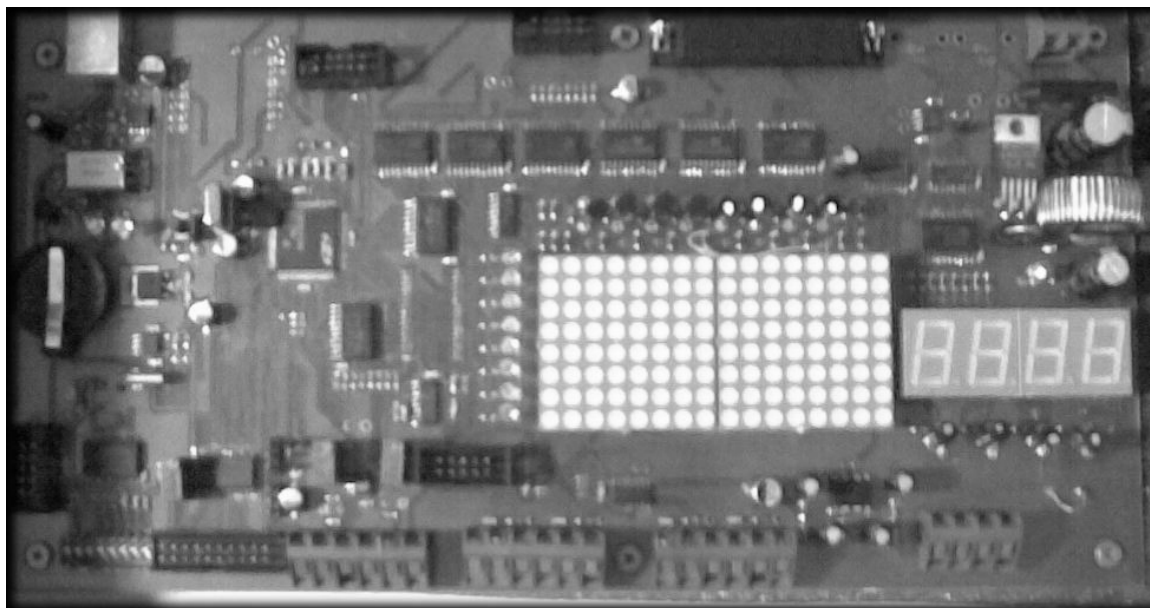


Рис.А.5. Фотография лабораторного стенда

Работа в среде Silicon Labs и работа с внутрисхемным отладчиком

Всплывающее меню File

Во всплывающем меню File рисунок А.6 доступны следующие опции: New File, Open File, Close File, Save, Save As, Save All, Print Setup, Print.

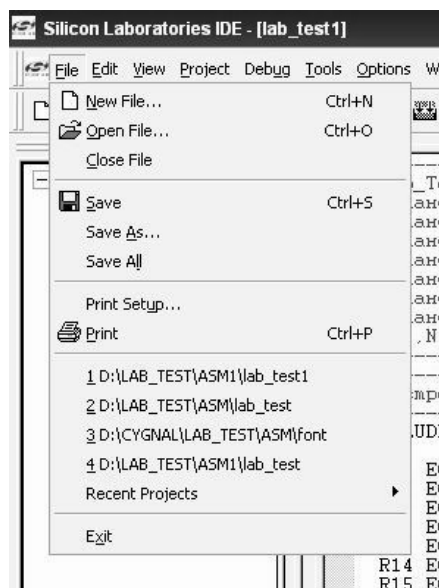


Рис.А.6. Меню File.

Эти опции предназначены для создания, сохранения, открытия и закрытия файлов.

Всплывающее меню View

Во всплывающем меню View рисунок А.7 доступны следующие опции: Debug Windows, Project Window, Output Window, Toolbars, Status bar, Workbook Mode.

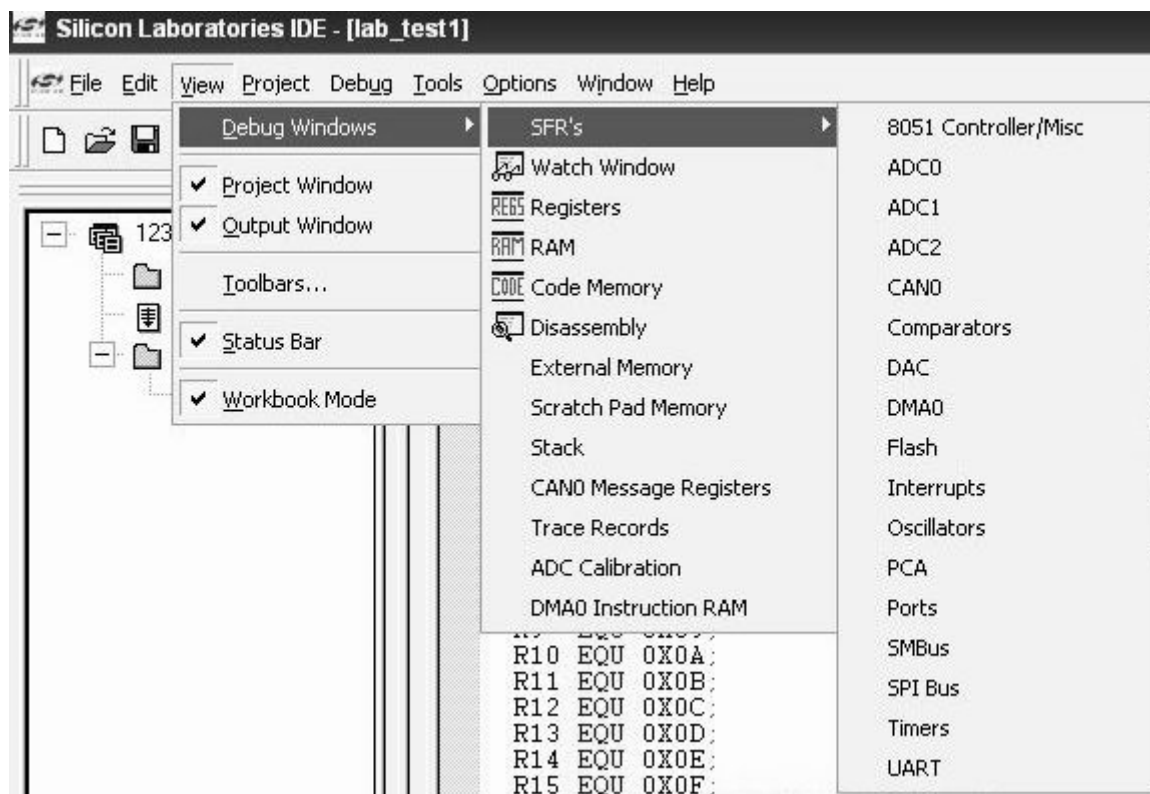


Рис.А.7. Меню View

Debug Windows. Для просмотра и отработки программы предусмотрены специальные окна, отображающие содержимое специальных регистров, памяти и т.д. Чтобы вызвать данные окна, нажимаем View>Debug Window:

- SFR's – окно регистров специального назначения;
- Watch Window – собственное окно просмотра;
- Registers – окно просмотра содержимого регистров;
- RAM – окно просмотра содержимого ячеек памяти;
- Code Memory – окно просмотра содержимого области памяти;
- Disassembly – окно, отображающее команды на языке ассемблер;

- External Memory – окно просмотра содержимого ячеек внешней памяти;
- Stack – окно просмотра содержимого Стека.

Создание нового проекта

1) Выберите из меню Project>New Project рисунок А.8.

В результате в «Окне проекта», находящегося слева, появится «корень дерева» проекта(new project), как показано на рисунке А.8.

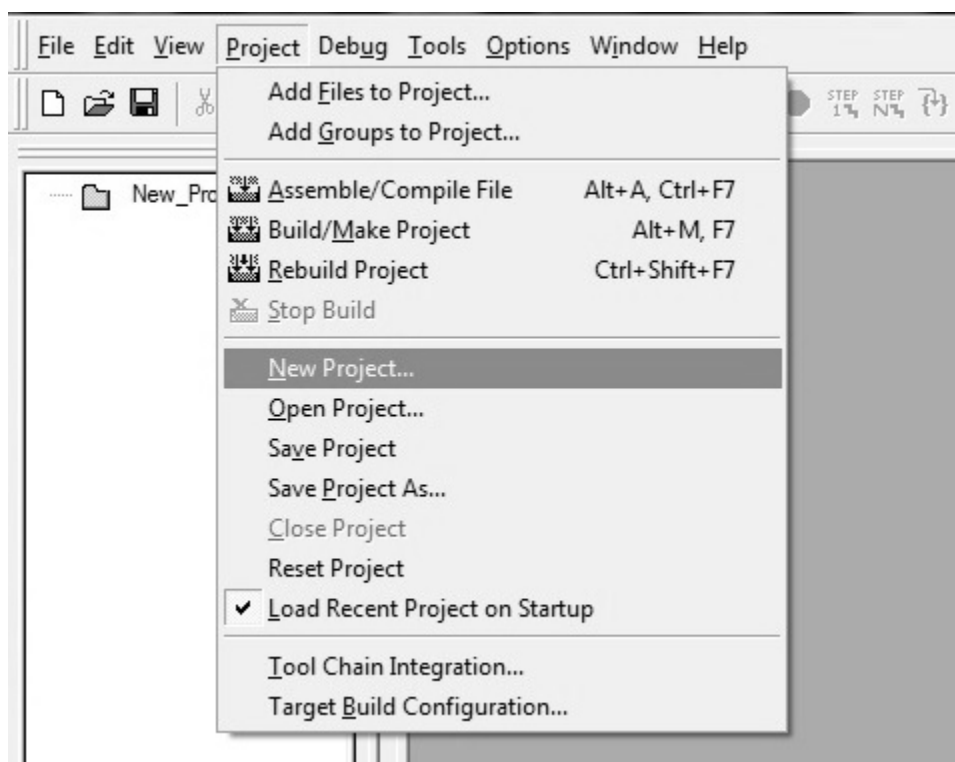


Рис.А.8. Создание нового проекта

2) Во всплывшем окне на вкладке Project в таблице Select Device Family выбираем нужный микроконтроллер C8051F06х, присваиваем имя проекта, место расположение. В Project Type для языка программирования «ASM» выбираем ASM Source Project рис. А.9.

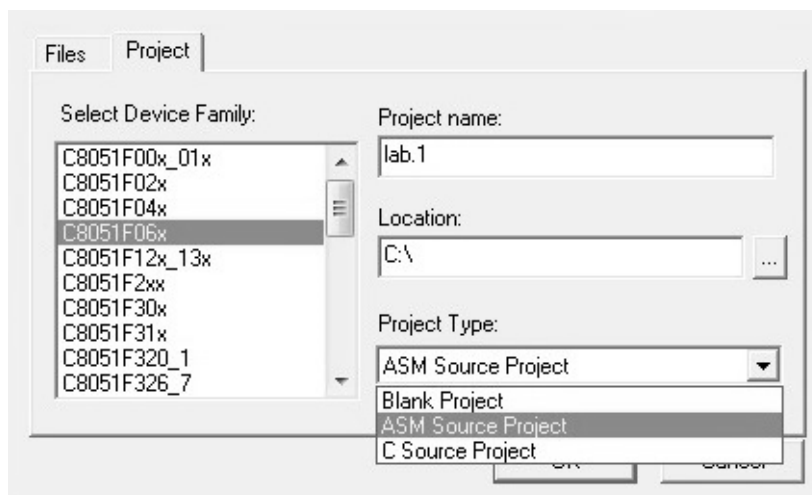


Рис. А.9. Выбор языка программирования «ASM»

а для языка программирования «C» необходимо выбрать C Source Project рис. А.10.

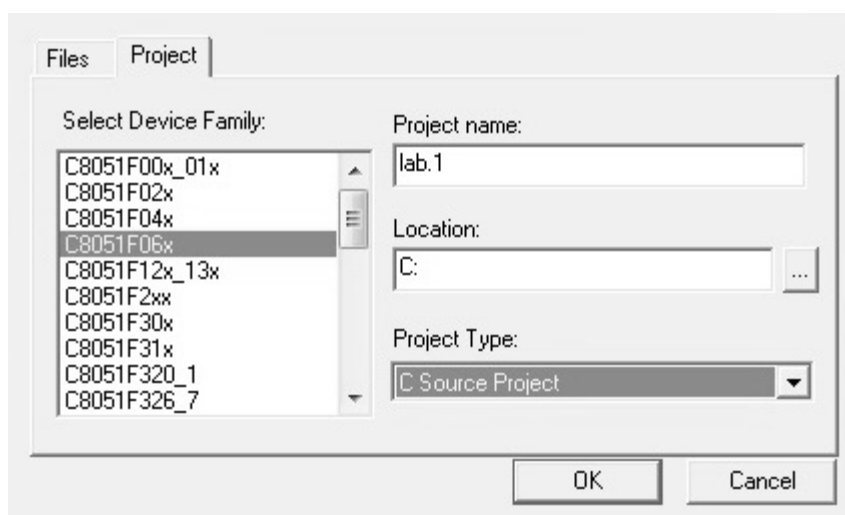


Рис. А.10. Выбор языка программирования «C»

- 3) Во вкладке Files присваиваем имя файлу, ставим галочки для добавления проекта Add to project и Add to build. Для программирования на языке «ASM» ставим галочку напротив ASM Source File как показано на рисунке А.11.

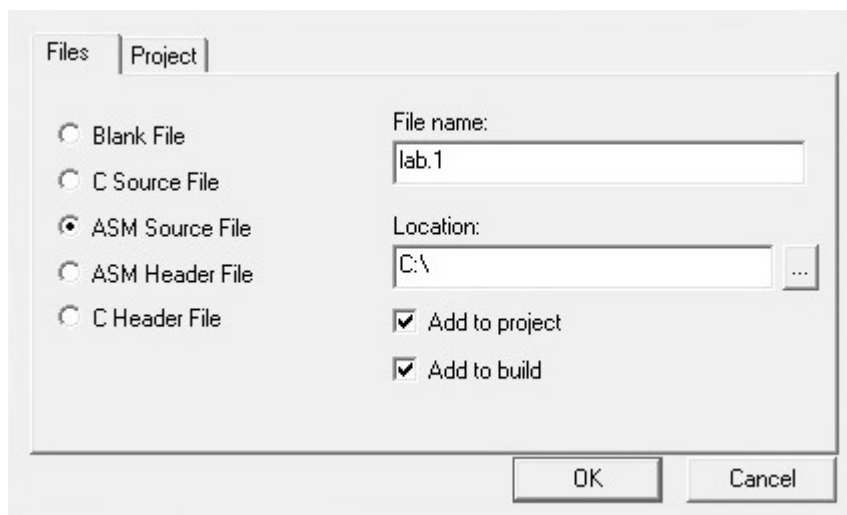


Рис. А.11. Создание файла на языке программирования «ASM»

Для программирования на языке «С» ставим галочку напротив C Source File как показано на рисунке А.12.

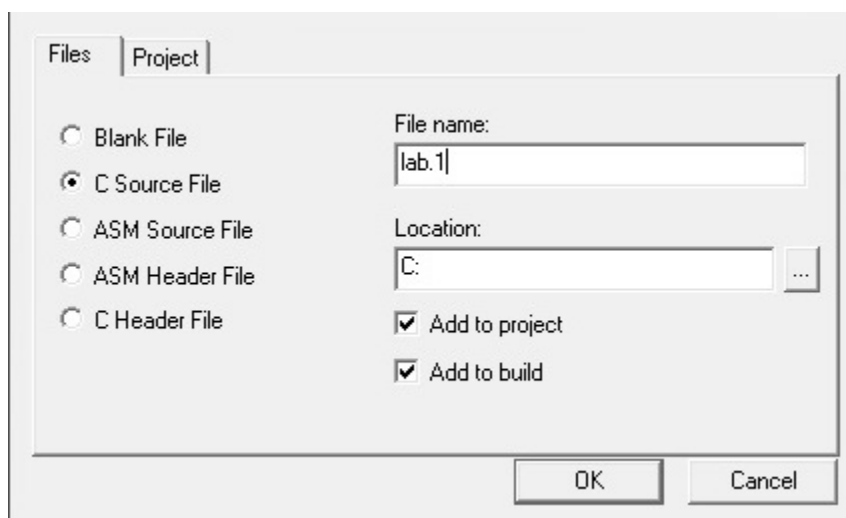


Рис. А.12. Создание файла на языке программирования «С»

Подключение лабораторного стенда к ПК.

Для подключения лабораторного стенда к ПК, необходимо физически соединить USB кабелем программатор лабораторного стенда и ПК. Программатор лабораторного стенда присоединиться с одним из виртуальных COM портов ПК, для определения номера COM порта заходим в Пуск – Панель управления – Система – Оборудование – Диспетчер устройств, раскрываем диалоговое окно Порты (COM и LPT),

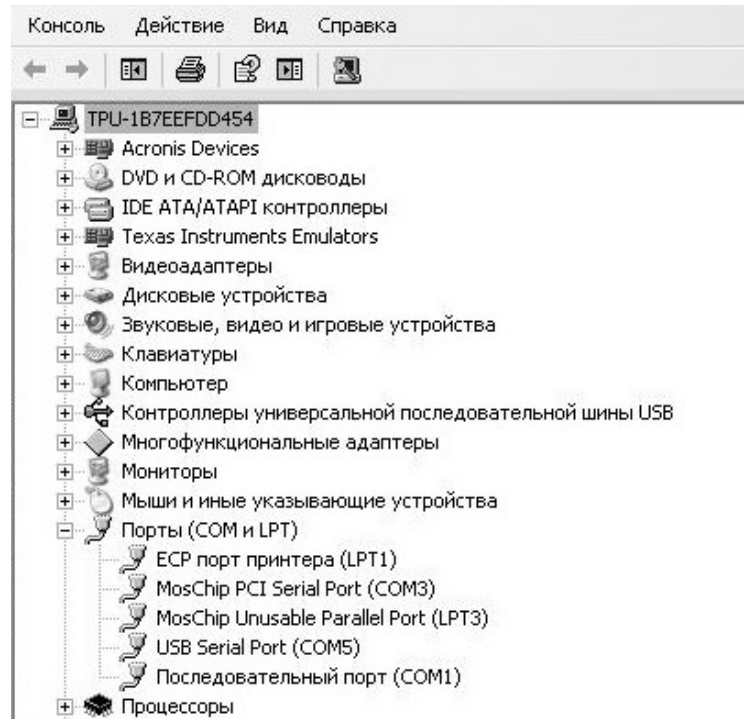


Рис. А.13. Диспетчер устройств.

наше устройство определяется как USB Serial Port (COMx), где “х” номер COM порта. Далее необходимо в Silabs прописать нужный COM порт, заходим в Options – Connection Options как показано на рис.А.14.

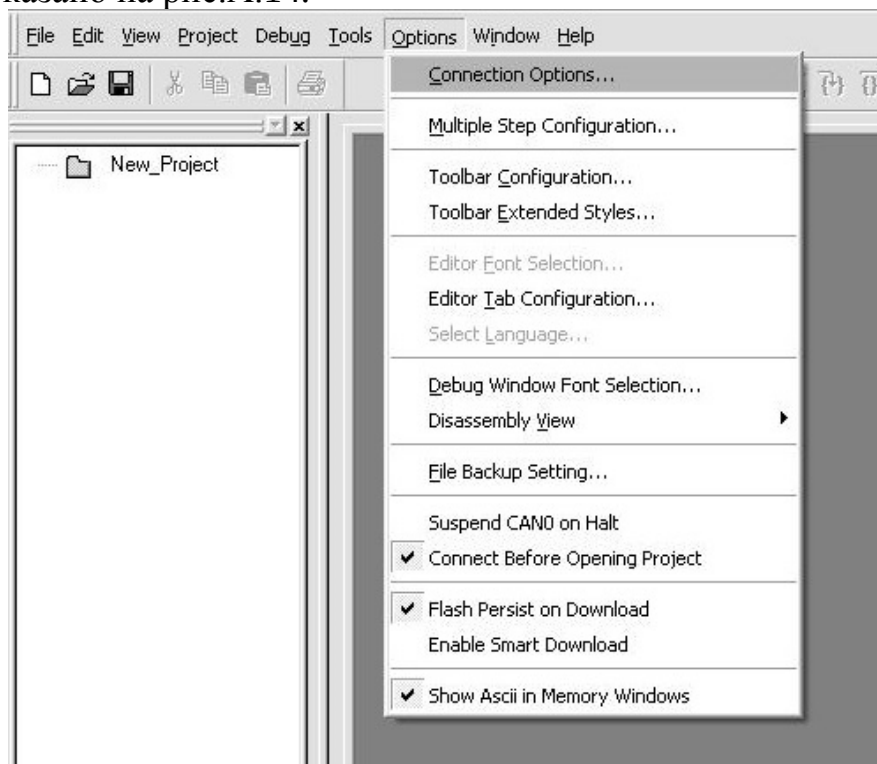


Рис. А.14. Вкладка Options.

Далее раскрываем COMPORT и из выпавшего списка выбираем виртуальный COM порт нашего устройства нажимаем кнопку «ОК» (рис.А.15).

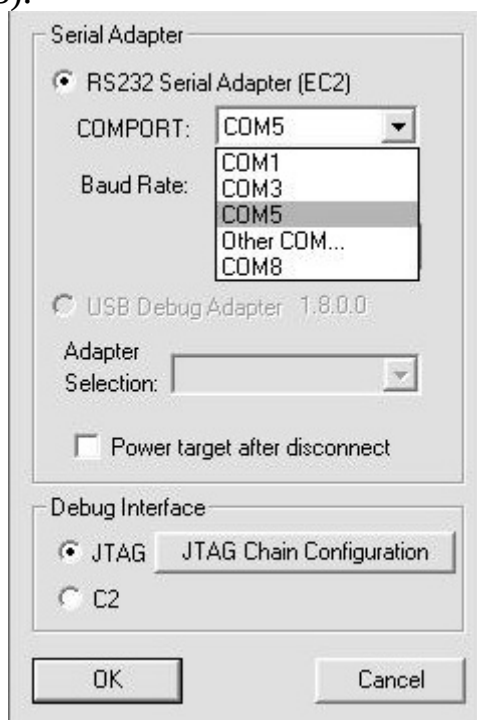


Рис. А.15. Выбор виртуального COM порта.

На панели программы нажимаем иконку Connect рис. А.16 г или набором клавиш (Alt+C), после этого связь лабораторного стенда с ПК будет установлена.

Функции необходимые для работы:

- Подключение Debug>Connect;
- Компиляция проекта Project>Assemble/Compile File;
- Пошаговое выполнение программы Debug>Step;
- Сброс контроллера Debug>Reset;
- Запуск выполнения Debug>Run.

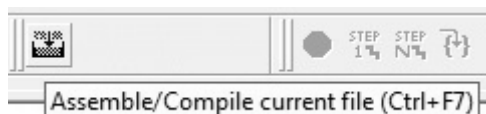
В контроллер можно загрузить только откомпилированный проект, не содержащий ошибок. Компиляция не произойдет, если на пути к каталогу, содержащему компилируемый файл, имеются русскоязычные символы.

Порядок работы при загрузке программы в контроллер:

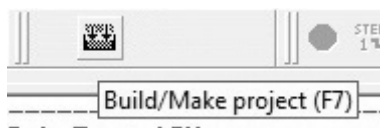
На рисунке А.16 показана панель программы, предназначенная для управления соединением программы с контроллером.

Assemble/Compile File (Ctrl+F7) – компиляция файла рис.А.16 а;

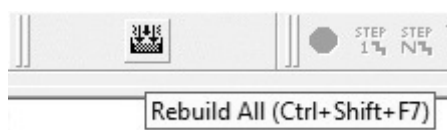
Build/Make Project (F7) – проверка проекта рис.А.16 б;
 Connect (Alt+C) – соединение с контроллером рис.А.16 в;
 Download code (Alt+D) – запись проекта в контроллер рис.А.16 д.



а)



б)



в)



г)



д)

Рис.А.16. Панель управления соединением

Поскольку компилировать программу приходится каждый раз, перед тем как загрузить в память контроллера, то кнопка загрузки в контроллер будет неактивна до момента успешного окончания компиляции.

Написав программу, можно посмотреть, как она функционирует. Для этого необходимо откомпилировать программу и записать в контроллер, после чего можно приступить к отладке. Внизу экрана располагается область, называемая Output Window. Она предназначена для сообщения пользователю о наличии или отсутствии ошибок в программе рисунок А.17.

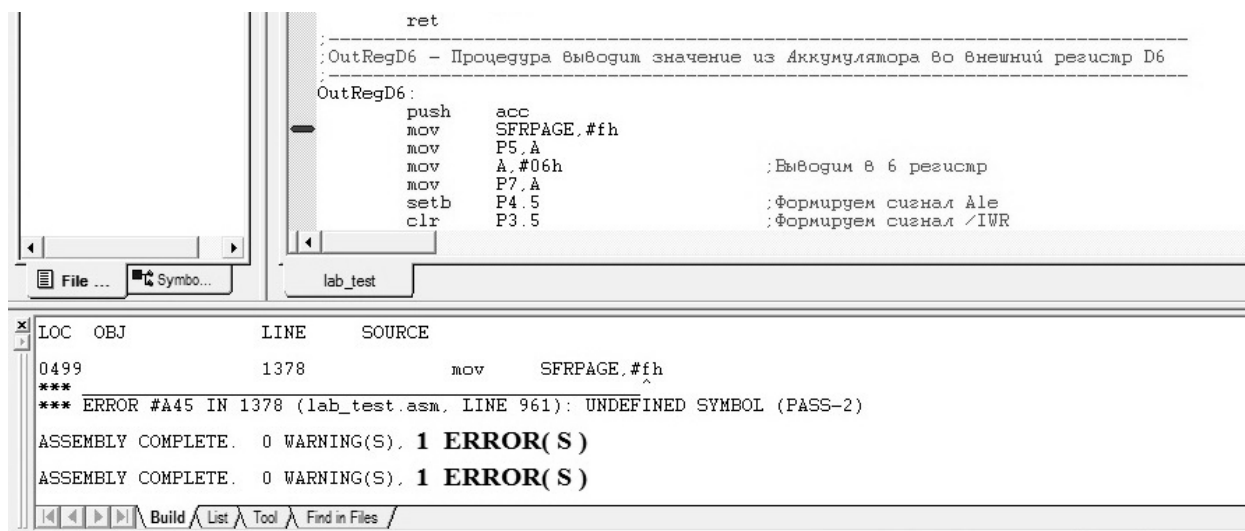


Рис.А.17. Окно ошибок

Это сообщение появится после компиляции проекта. Если программа содержит ошибки рис.А.17, то в сообщении будет указано, сколько ошибок содержит проект, в какой строке допущена ошибка, и что именно неверно. В данном примере не определился символ «f», при использовании шестнадцатеричной системы счисления перед буквенными символами ставится цифра «0».

Пример: MOV R0, #0FFH

Цифра «0», говорит о том, что «FF» это числа, «H» Hex(анг.)-шестнадцатеричной системы счисления.

Для отладки следует открыть (Debug Window) и производить «манипуляции» с программой.

Для этого имеется специальное средство, называемое Breakpoint, то есть точка останова. Для того чтобы поставить её на какой-нибудь строке программы, нужно нажать на ней правой кнопкой мыши и выбрать Insert/Remove Breakpoint рисунок А.18.

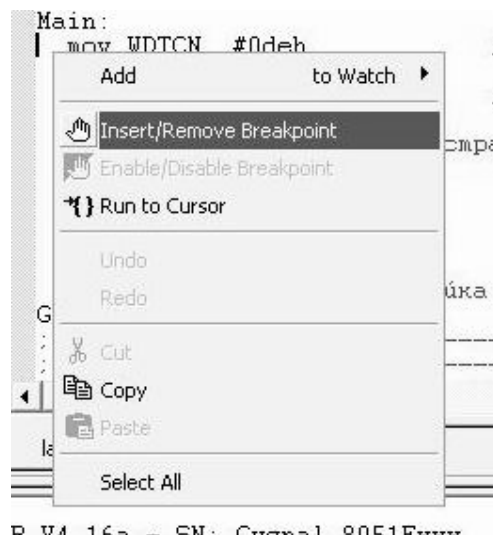


Рис.А.18. Выбор точек останова программы

Если точка останова была установлена, то на поле слева появится значок, показанный на рисунке А.19, означающий, что на этом месте программа остановится, и будет ждать дальнейших указаний.

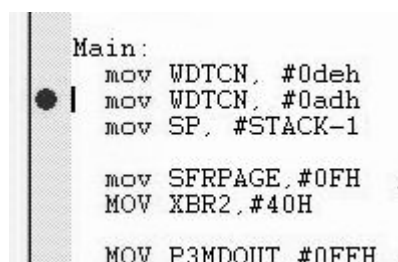


Рис.А.19. Значок точки останова программы

Для того, чтобы снять эту точку, необходимо ещё раз проделать данную процедуру.

Стрелка синего цвета рисунок А.20, располагаемая на том же поле, что и Breakpoint, указывает на ту строчку, которая будет выполнена на следующем шаге.



Рис.А.20. Индикатор выполнения следующей команды, после остановки программы

Также в отладочном средстве имеется возможность указывать выполнение программы до определенной строки без установки Breakpoint. Это может оказаться полезным в том случае, если необходима разовая проверка выполнения программы до этой строки. В этом случае, на нужной строке нажимаем правой кнопкой мыши и выбираем Run To Cursor рисунок А.21.

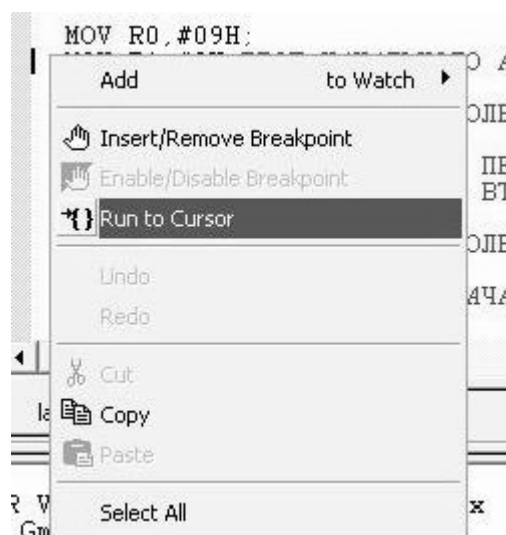


Рис.А.21. Выбор разовой точки останова

Программа выполнится до указанной строки и остановится. После этого можно пронаблюдать содержимое регистров и т.д., чтобы удостовериться в правильности исполнения программой поставленной задачи. Когда вся работа будет выполнена, можно снова запустить программу. Она продолжит выполнение с того места, на котором была остановлена.

ПРИЛОЖЕНИЕ Б

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ СИ

Алфавит языка СИ

В алфавит языка СИ входят:

- О латинские буквы: от а до z и от А до Z;
- О десятичные цифры: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9;
- О специальные символы: { } [] (), % " | / \ ; ' : ? < = > _ ! & # ~ ^ . * + -

К специальным символам относится также пробел. Буквы кириллического алфавита могут использоваться в комментариях и символьных константах.

Некоторые символы, если они не разделены пробелом, идентифицируются как один символ, например:

++ -- == && || «»>=<+=-*=/=?: /**/

В качестве ограничителей комментариев используются пары символов /* и */, а также строковой комментарий в виде //.

Например:

/ Многострочный комментарий для языка программирования СИ*
*/

// Строковый комментарий для СИ

Из символов алфавита формируются единицы текста программы (лексемы), которые при компиляции воспринимаются, как единое целое и не могут быть разделены на более мелкие элементы. К лексемам относятся идентификаторы, служебные слова, константы, знаки операций, разделители.

Идентификаторы языка СИ

Идентификатор – это последовательность латинских букв, цифр, символов подчеркивания, начинающаяся с буквы или символа подчеркивания.

Например:

X12	metka	Metka_2	bad_call_3
-----	-------	---------	------------

В СИ различаются прописные и строчные буквы. Например, metka, Metka, mEtka – разные идентификаторы.

Существуют ограничения на длину идентификатора, которые могут различаться в разных реализациях языка. Обычно два идентификатора считаются различными, если у них различаются первые 32 символа имени.

Например,

Ab1234567890123456789012345678901

и

Ab123456789012345678901234567890d - это одна и та же переменная, поскольку у нее одинаковые первые 32 символа.

В некоторых реализациях компиляторов допускаются идентификаторы длиной не более 8 символов.

Служебные слова

Служебные слова в СИ - это идентификаторы, назначение которых однозначно определено в языке. Полный список служебных слов зависит от используемого компилятора. Их употребление строго определено, и они не могут использоваться иначе. Служебными словами языка СИ являются:

<i>auto</i>	<i>extern</i>	<i>sizeof</i>
<i>break</i>	<i>float</i>	<i>static</i>
<i>case</i>	<i>for</i>	<i>struct</i>
<i>char</i>	<i>goto</i>	<i>switch</i>
<i>const</i>	<i>if</i>	<i>typedef</i>
<i>continue</i>	<i>int</i>	<i>union</i>
<i>default</i>	<i>long</i>	<i>unsigned</i>
<i>do</i>	<i>register</i>	<i>void</i>
<i>double</i>	<i>return</i>	<i>volatile</i>
<i>else</i>	<i>short</i>	<i>while</i>
<i>enum</i>	<i>signed</i>	

Дополнительные служебные слова используются в конкретных реализациях языка СИ. Некоторые начинаются с подчеркивания, например: `_myfun`, `^abc` и др. Кроме того, существуют служебные слова, которые начинаются с двойного нижнего слеша: `__myfun`.

Поэтому не рекомендуется использовать в программе идентификаторы, начинающиеся с подчеркиваний, во избежание совпадения их со служебными словами.

Типы данных

Данными языка СИ являются переменные и константы. Переменные меняются во время выполнения программы, а константы остаются неизменными.

Данные подразделяются на определенные типы. Под типом данных понимают множество допустимых значений, множество разрешенных операций и размер памяти, занимаемый данными этого типа.

Задать тип данных означает сообщить компилятору:

1. имя переменной или константы;
2. ее начальное значение;
3. размер памяти, необходимый для хранения ее значений;
4. вид и способ выделения памяти;
5. какие действия можно выполнять с переменной или константой.

В языке СИ все переменные должны быть объявлены до их использования. В СИ определены 5 базовых типов данных:

- char – символьный;
- int – целочисленный;
- float – с плавающей точкой;
- double – с плавающей точкой двойной длины;
- void – пустой, не имеющий значения.

char и int являются целочисленными типами данных и хранят целые числа. Несмотря на то, что char – это символьная переменная, в типе char хранится информация о коде символа из ASCII таблицы.

Переменные типа float и double являются числами с плавающей точкой.

На основе этих пяти видов строятся других типы данных с использованием модификатора, который ставится перед соответствующим типом данных.

В стандарте ANSI языка СИ такими модификаторами являются следующие служебные слова:

signed – знаковый;
unsigned – без знаковый;
long – длинный;
short – короткий.

В таблице 1 представлены арифметические типы данных, указан объем занимаемой памяти и диапазон допустимых значений

Таблица 1

Тип данных	Размер в байтах	Диапазон значений	Эквивалентные названия
char	1	-128...+127	signed char
int	2/4	зависит от системы	signed, signed int

unsigned char	1	0...255	нет
unsigned int	2/4	зависит от системы	unsigned
short int	2	-32768...32767	short, signed short int
unsigned short	2	0...65535	unsigned short int
long int	4	- 2147483648...2147483647	long, signed long int
unsigned long int	4	0...4294967295	unsigned long
float	4	$\pm(3.4E-38...3.4E+38)$	нет
double	8	$\pm(1.7E-308...1.7E+308)$	нет
long double	10	$\pm(3.4E-4932...3.4E+4932)$	нет

Исходя из таблицы 1, можно сделать следующие выводы:

- если не указан базовый тип, то по умолчанию это int;
- если не указан модификатор, то по умолчанию это signed;
- с базовым типом float модификатор не используется.

Различие между целыми числами со знаком и целыми числами без знаков состоит в интерпретации старшего бита целого числа. Если старший бит равен нулю, то число положительное, а если равен единице, то число будет отрицательным.

Например, целое число +21 в памяти компьютера будет храниться в виде:

0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1

Объявление переменных в программе

Описание переменных в программах на СИ/СИ++ имеет вид:

<i>имя_типа список переменных;</i>

При объявлении переменной компилятор выделяет место в памяти компьютера, необходимое для размещения переменной указанного типа.

Примеры объявления переменных:

```
int a,b,c;
float radius, visota, dlina;
char symbol,cc;
```

```
unsigned char code;  
unsigned long long_number;
```

Одновременно с объявлением переменной можно задать начальные значения переменных, т.е. произвести инициализацию переменной. Инициализация и описание переменной имеет следующий общий вид:

<i>имя_типа список переменных=начальное_значение;</i>

Например,
float pi=3.14159,c=1.23;
unsigned int month=2000;

При объявлении переменных имеет очень большое значение место объявления этой переменной, так называемое правило видимости.

В языке СИ переменная может быть объявлена в трех местах:

1. Вне каких либо функций, в том числе и `main( )`. Такая функция называется глобальной и может быть использована в любом месте программы.
2. Внутри тела функции или блока. Такая функция называется локальной и может быть использована только внутри этого блока, для других блоков программы она неизвестна
3. Как параметр функции, при этом переменная используется для передачи информации этой функции. Такая переменная называется локальная переменная функции.

Примеры объявления переменных в разных местах программы.

```
#include<conio.h>  
#include<stdio.h>  
char a; /*а объявлена глобальной переменной*/  
main()  
{  
    int x; /* x объявлена локальной переменной для функции main*/  
    printf("Введите символ для повторения:");  
    a=getche();  
    printf("Введите количество повторения символа в строке");  
    scanf("%d",&x); /*использование локальной переменной*/  
    print_stroku(x); /*обращение к функции символьной печати строки*/  
}
```

```

print_stroku(int y) /*заголовок функции и объявление формального
параметра y*/
{
    int j; /*объявление локальной переменной для функции
print_stroku*/
    for(j=0;j<y;j++) /*использование локальной переменной*/
        printf("&c\n",a); /*использование глобальной переменной*/
}

```

При написании программы необходимо знать следующие правила:

1. глобальные переменные, объявленные в программе, не могут иметь одинаковые имена.
2. Имя локальной переменной должно быть уникальным только внутри функции, в котором она объявлена.
3. В одном блоке локальные имена не могут иметь одинаковых имен, равно как и формальный параметр функции, используемый в этом блоке.

Объявление констант в программе

Константы – это фиксированные величины, значения которых не может быть изменено в ходе выполнения программы. Тип данных констант может быть любой из базового типа данных.

Например,

```

int    1, 10, -258;
char   'x', 'a', '78';
float  3.14, 2.57E-27;

```

При объявлении констант необходимо соблюдать правила.

Целочисленная константа относится к типу `int`, если она входит в ее интервал типа `int`, размер которого зависит от системы, иначе она считается типа `unsigned`. Если же не она не входит и в интервал `unsigned`, то автоматически она считается типа `long`.

Константа считается типа `double`, если она с десятичной точкой и входит в интервал типа `double`.

Очень часто при задании констант используются суффиксы. Для целочисленных констант – `u`, `l`, `h`, `U`, `L`, `H`, для с плавающей точкой – `l`, `L`, `f`, `F`.

При программировании также зачастую используются константы в восьмеричной и шестнадцатеричной системах исчисления. Перед шестнадцатеричной константой ставится `0x`, а восьмеричная константа всегда начинается с нуля.

Запись числа в различных системах счисления

Десятеричное число	Двоичное число	Шестнад- цатеричное число	Восьмерич- ная константа	Шестнад- цатеричная константа
0	0000	0	00	0x0
1	0001	1	01	0x1
2	0010	2	02	0x2
3	0011	3	03	0x3
4	0100	4	04	0x4
5	0101	5	05	0x5
6	0110	6	06	0x6
7	0111	7	07	0x7
8	1000	8	10	0x8
9	1001	9	11	0x9
10	1010	A	12	0xA
11	1011	B	13	0xB
12	1100	C	14	0xC
13	1101	D	15	0xD
14	1110	E	16	0xE
15	1111	F	17	0xF

Кроме численных констант могут задаваться также строковые константы. Строковая константа – это набор символов, заключенных в двойные кавычки.

Для хранения строковой константы требуется $n+1$ байтов, где n – число символов между кавычками. Дополнительный последний байт требуется для хранения кода 000, который записывается автоматически.

Не следует путать строковую и символьную константы.

Символьные константы бывают клавиатурные и кодовые.

- О клавиатурные – это клавиатурный символ в апострофах: '1', 'i', 'y';
- О кодовые – это управляющие символы в апострофах: '\n', '\b', '\t', '\\';

\n	новая строка	\r	возврат каретки
\t	горизонтальная табуляция	\f	протяжка формата
\b	шаг назад (backspace)	\\	обратный слеш
\'	апостроф	\"	кавычки
\a	звук	\v	Вертикальная табуляция

Операции, операторы и выражения

Операции задают определенную последовательность действий и содержат знаки операций и операнды.

Операции могут проводиться с одним, двумя и тремя операндами. Соответственно, различают унарные, бинарные и тернарные операции.

Операнд представляет собой элемент операции. Операндами могут быть константы, переменные, и выражения.

Основные арифметические операции.

Примером унарной арифметической операции является операция изменения знака числа, обозначаемая символом «минус», который стоит перед одним операндом: $-x$.

Примером бинарной операции может служить простое выражение вычитания: $3.4 - x$.

Примером тернарной операции может служить условная операция, которая имеет следующую форму записи:

(условие)?значение«истина»:значение«ложь»

и содержит три операнда: условие, значение«истина» (символизируется знаком?) и значение«ложь» (символизируется знаком :).

Если условие выполняется, то результатом будет значение«истина», если не выполняется то результатом будет значение«ложь». Например, $b = (a > 0) ? a : 0$. Если $a > 0$, то $b = a$, иначе $b = 0$.

$+$, $-$, $*$, $/$ знаки бинарных операций сложения, вычитания, умножения и деления.

Имеется две разновидности операции деления: *целочисленное деление* (деление с остатком), когда оба операнда целые числа, и результат тоже целое число и *деление без остатка* (результат - вещественное число). Например, $5/2 = 2$, а $5.0/2 = 2.5$.

Для целых чисел определена операция $\%$ — деление по модулю.

Например, $8\%3$ равно 2.

Символ $=$ обозначает бинарную операцию простого присваивания, при которой значение правого операнда присваивается левому операнду, например: $v = 0.5 * a * t * t + 7$

Допускается *множественное присваивание*: $z = y = x = t = 68$.

В результате переменные z , y , x , t примут значение 68, если все они одного типа.

Выражение — это последовательность знаков операций, операндов и круглых скобок, которая задает вычислительный процесс получения результата определенного типа. Простейшим выражением является константа, переменная или вызов функции.

Вызов функции представляет собой имя функции, за которым в круглых скобках указывается список аргументов (возможно пустой). Примеры выражений с вызовом функций:

$$a*\sin(x+k) + b*\cos(x-k).$$

Можно считать, что во время выполнения программы результат, возвращаемый вызванной функцией, заменяет вызов функции.

Также как результат вычисления выражения заменяет само выражение. Например, выражение

$$-2-(\cos(x)+5*y) \text{ заменится значением } -8.0, \text{ если } x = 0 \text{ и } y = 1.$$

Порядок вычисления выражения определяется расположением знаков операций, круглых скобок и *приоритетами* выполнения операций.

Например: $(11+25/5+7)$ не равно $(11+25)/(5+7)$.

Операции с наивысшим приоритетом выполняются первыми. Приоритеты операций приведены в Таблице 4..

Таблица 4 Приоритеты и порядок выполнения операций

Лексемы	Операция	Приоритет	Ассоциативность
имена, литералы	простые лексемы	16	нет
m[i]	индексы	16	слева направо
f(...)	вызов функции	16	слева направо
.	прямой выбор	16	слева направо
->	опосредованный выбор	16	слева направо
(имя типа) {init}	составной литерал	15	справа налево
++ --	Инкремент и декремент	15	справа налево
sizeof	размер	15	справа налево
~	побитовое НЕ	15	справа налево
!	логическое НЕ	15	справа налево
- +	изменение знака, плюс	15	справа налево
&	адрес	15	справа налево
*	опосредование (разыменование)	15	справа налево
(имя типа)	приведение типа	15	справа налево
*, /, %	мультипликативные	13	слева направо

	операции		
+, -	аддитивные операции	12	слева направо
<<, >>	сдвиг влево и вправо	11	слева направо
<, >, <=, >=	отношения	10	слева направо
==, !=	равенство/неравенство	9	слева направо
&	побитовое И	8	слева направо
^	побитовое исключающее ИЛИ	7	слева направо
	побитовое ИЛИ	6	слева направо
&&	логическое И	5	слева направо
	логическое ИЛИ	4	слева направо
?:	условие	3	справа налево
=, +=, -=, *=, /=, %=, <<=, >>=, &=, ^=, =	присваивание	2	справа налево
,	последовательное вычисление	1	слева направо

В языке СИ предусмотрены две унарные операции: инкремента ++ и декремента -- соответственно для увеличения и уменьшения значения операнда на единицу. Например: `time++`; означает: `time = time+1`.

Их можно указывать перед операндом (++x или --x, префиксный инкремент и декремент), тогда значение операнда x изменяется перед его использованием и после операнда (x++ или x-- постфиксный инкремент и декремент) тогда значение x изменяется после его использования.

Например:

`b=3;`

`c=5;`

`a=b+c++;`

это означает, что `b=3; a=3+5=8; c=5+1=6;`

`b=3;`

`c=5;`

`a=(b++)+c;`

значит `b=4; c=5; a=4+5=9`.

В логических выражениях используются знаки операций отношения: меньше <, больше >, равно (совпадает) ==, меньше или равно <=, больше или равно >=, не равно !=.

Запятая - знак специальной операции задающей последовательность выражений. Например, выражение (i=1, j=4, k=7) состоит из трех простых выражений.

Круглые скобки () используются для изменения очередности выполнения операций.

Операция sizeof вычисляет размер операнда в байтах и имеет две разновидности: sizeof(тип) и sizeof выражение. Например: sizeof(int) или sizeof Mytext. Здесь Mytext может быть именем строковой константы.

Для компактной записи операторов присваивания используются составные операции присваивания: *=, /=, %=, +=, -=, <<=, >>=, &=, ^=, |=

Например, при выполнении двух операторов a=a+b; или a+=b; результат будет одинаков.

Если в выражении содержится несколько операций одного приоритета на одном и том же уровне, то их обработка производится в соответствии с порядком выполнения: справа налево или слева направо.

Например: при

x = 4, a = 5;

$x/2 + a * 5 \% 15 = 12 \Rightarrow 4/2 + 5 * \text{остаток от деления}(25 \% 15) \Rightarrow 2 + 10 = 12;$

$x/2 + a * (5 \% 15) = 27 \Rightarrow 4/2 + 5 * (\text{остаток от деления}(5 \% 15)) \Rightarrow 2 + 5 * 5 = 27$

Операторы в отличие от выражений, являются законченными предложениями языка. Они могут содержать несколько выражений, разделенных запятой.

Обычно операторы заканчиваются точкой с запятой. Исключение составляют:

- директивы препроцессора, начинающиеся с символа #,
- тела функций,
- блоки операторов.

Простые операторы с помощью фигурных скобок {} можно объединить в блок, который называется составным или сложным оператором и за которым точку с запятой можно не ставить.

Например,

```
{  
    a=5*i-1;  
    printf("\n a = ",a);  
}
```

составной (сложный)
оператор

Операторы имеют названия, отражающие их назначение или выполняемые действия.

В языке СИ существует 5 основных видов операторов:

1. *операторы объявления*, в которых объявляются типы данных, константы, переменные и заголовки функций. Например: `int a, b;`
2. *операторы присваивания*, в которых используется операция присваивания. Например: `y=(a*x+b)*x+c;`
3. *операторы вызова функции*, в которых вызывается функция. Например: `rectangle(x1,y1,x2,y2);`
4. *операторы управления*, с помощью которых осуществляется управление работой программы.

Например: `if`, `switch`, `break`, `continue`, `return` или **операторы цикла**;

5. *пустые операторы*, которые не выполняет никаких действий и состоят из одной только точки с запятой. Например: `;`.

Программирование разветвлений

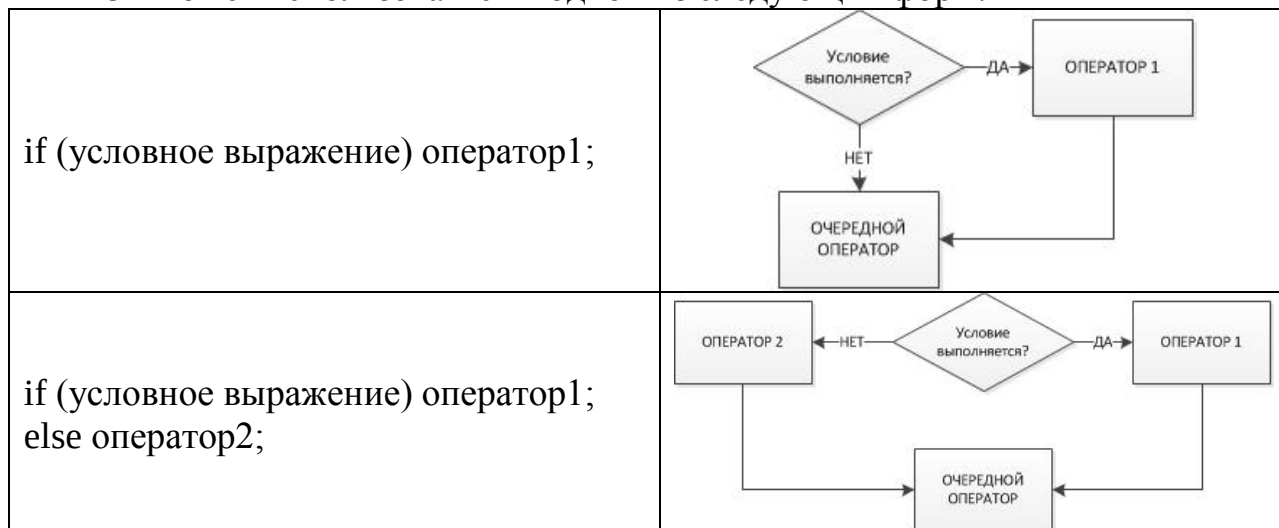
Для программирования разветвлений в языке СИ предназначены:

- условный оператор (`if...else`),
- операция условия (`?:`),
- оператор переключатель (`switch`).

Условный оператор if

Условный оператор обеспечивает выполнение или невыполнение некоторого оператора или группы операторов в зависимости от заданного условия, изменяя естественный порядок выполнения программы.

Он может использоваться в одной из следующих форм:



Если значение условного выражения истинно (отлично от нуля), то выполняется оператор1; если ложно (равно нулю), то для первой формы оператор1 пропускается, а для второй формы после пропуска оператора1 выполняется оператор2, стоящий после слова else. Если после проверки условия необходимо выполнить несколько операторов, то они заключаются в блок с помощью фигурных скобок {}.

Пример 1. Вычислить переменную «с», используя оператор if.

<pre>... int a=1, b=2, c; if(a<b)a=a+b; /*один оператор*/ c=a+b; ...</pre>	<pre>... int a=1, b=2, c; if(a<b) {a=a*a; b=b+10;} /*блок операторов*/ c=a+b; ...</pre>
---	--

Пример 2. Оператора if...else Найти минимум из двух чисел x и y.

```
...
int x, y, m;
if(x<y) m=x;
else m=y;
cout << "minimum = " << m;
```

Иногда нужно объединить два или более простых условных выражения в одно условие. Для этого используются *логические операции*.

Логические операции:

&& – логическое И;

|| – логическое ИЛИ;

! – логическое НЕ.

Пусть expr1 и expr2– два простых условных выражения. Тогда:
 выражение expr1 && expr2 истинно тогда и только тогда, когда истинны оба – expr1 и expr2;
 выражение expr1 || expr2 истинно когда, когда истинно хотя бы одно из двух expr1 или expr2;
 выражение !expr1 истинно тогда, когда expr1 ложно и наоборот.

Например:

5>2&&7>4 истинно; 5>2&&7<4 ложно;
5>2||7<4 истинно; 5<2&&7<4 ложно;
!(7<4) истинно; !(7>4) ложно.

Операторы if могут быть вложены друг в друга.

Пример 3. Использование вложенных операторов if...else
Отыскание максимума из трех чисел a, b, c.

```
...  
int a, b, c, m;  
if(a>b&& a>c) m=a;  
else if(b>c) m=b;  
else m=c;  
cout << "maximum=" << m;  
...
```

Операция условия ?:

Для короткой записи оператора if...else в языке СИ используют операцию условия.

Пример 5. Найти максимум из двух чисел x и y.

```
...  
int x, y, m;  
m=(x>y) ? x : y;  
cout << "maximum = " << m;  
...
```

Применение операции условия не является обязательным, так как тех же результатов можно достичь при помощи оператора if...else, однако при ее использовании получаются более компактные выражения и **более компактный машинный код**.

Множественный выбор: операторы switch и break

Операция условия и условный оператор if...else позволяют легко осуществить выбор между *двумя* вариантами.

Однако иногда возникает необходимость осуществить выбор одного варианта из *нескольких*.

Это можно сделать, используя вложенные условные операторы. При этом сама программа становится более сложной, и менее понятной.

Для выбора одного из множества вариантов служит оператор-переключатель switch, который имеет следующую форму записи:

```
switch(выражение)  
{  
case константа1: оператор1; break;  
case константа2: оператор2; break;
```

```

...
case константаN: операторN; break;
[default: оператор default; break;]
}

```

Оператор выбора работает следующим образом. Сначала вычисляется выражение, стоящее в скобках после слова switch. Затем осуществляется переход на ту из меток, обозначенную словом case, значение константы, у которой совпало со значением выражения в скобках после switch.

Константа, стоящая после case, должна быть целого типа.

Если вычисляемое выражение не совпало ни с одной из проверяемых констант, то осуществляется переход на необязательную метку default. Если нет метки default, то управление передается оператору, следующему за оператором switch. Рекомендуется для поиска ошибок включать метку default, даже тогда, когда учтены все возможные случаи.

Каждая ветвь заканчивается оператором break, выполнение которого приводит к выходу из оператора switch. Если break отсутствует, то управление передается следующему оператору, помеченному case или default.

Ключевые слова case и default не могут находиться за пределами блока switch.

Пример 6. Применение оператора switch.

Требуется напечатать название месяца в зависимости от его номера – переменной month.

```

...
switch(mesyaz)
{
case 1: cout << "\n January"; break;
case 2: cout << "\n February"; break;
case 3: cout << "\n March"; break;
...
case 12: cout << "\n December"; break;
default: cout << "\n Month is wrong";
}
...

```

Операторы циклов

Цикл представляет собой участок программы, в котором одни и те же вычисления неоднократно повторяются над различными значениями одних и тех же переменных.

Для организации циклов в Си используются следующие три оператора: while, for и do-while.

Цикл while

Цикл while является циклом *с предусловием*. Он используется в случае, когда, во-первых, не известно точное число повторов и, во-вторых, нет необходимости, чтобы цикл непременно был выполнен хотя бы один раз. Цикл while имеет следующую форму записи:

while(условие) тело цикла;

На месте тела цикла, может стоять простой оператор или совокупность операторов, объединенных в блок скобками {}.

Если условие истинно, то тело цикла выполняется один раз, затем условие проверяется заново. Итерации выполняются до тех пор, пока условие не станет ложным. В тело цикла while должны быть включены **конструкции, изменяющие проверяемое условие** так, чтобы в конце концов оно стало ложным. Иначе выполнение цикла станет бесконечным.

Пример 7. Цикл while. Пользователю дается 10 попыток для угадывания заданного числа.

```
...
i=1; rez=1;
while(i++<=10&&rez!=25)
{
    cout << "\nВведите число:";
    cin >> rez;
}
if(i==12)cout << "\nВы не угадали.";
else cout << "\nПоздравляю! Вы угадали число.";
```

В данном примере цикл выполняется до тех пор, пока не угадано число 25 или не исчерпано количество попыток.

Цикл for

Цикл for является циклом *с параметрами* и обычно используется, когда известно точное количество итераций. При этом осуществляются три операции: *инициализация* счетчика цикла, *сравнение* счетчика цикла с граничным значением и *коррекция* счетчика при каждом прохождении тела цикла.

В принципе эти действия можно реализовать с помощью цикла while. Инициализацию счетчика можно выполнить за пределами цикла перед его выполнением, приращение счетчика можно осуществить в теле цикла, но здесь есть опасность не проинициализировать счетчик или не выполнить его приращение. Это ведет к повышению вероятности возникновения ошибок и затрудняет отладку.

В цикле for данные три действия собраны вместе, что позволяет избежать указанных неприятностей. Цикл for имеет следующую форму записи:

for(инициализация счетчика; проверка условия; коррекция счетчика)
 тело цикла;

Инициализация счетчика цикла вычисляется один раз, когда цикл for встречается в программе. Затем *проверяется условие*. Если условие истинно, то выполняется тело цикла. Если ложно, то цикл завершается. В конце каждого выполнения тела цикла счетчик цикла *корректируется*.

Телом цикла может быть простой оператор или блок операторов, заключенных в скобки {}.

В круглых скобках после слова for могут отсутствовать все выражения или любое из них, но должны обязательно быть две точки с запятой.

Специальное правило для цикла for гласит, что если отсутствует условное выражение, то результат проверки условия всегда истина.

Пример 8. Пример бесконечного цикла

```
...  
for(;;);  
...
```

Пример 9. Цикл for. Пусть требуется вычислить y^{10} . Возможное решение имеет вид:

```
...  
for(i=1, rezult=1; i<=10; i++)  
    rezult=rezult*y;  
cout << "\n y's 10-th power =" << rezult;
```

Цикл do-while

Цикл do-while является циклом *с постусловием* и используется в тех случаях, когда неизвестно точное количество повторов, но в то же время цикл необходимо выполнить по меньшей мере один раз. Цикл do-while очень похож на цикл while; разница состоит только в том, что

проверка условия в цикле do-while происходит после выполнения тела цикла. Этот цикл имеет следующую форму записи:

do тело цикла; while(условие);

Пример 10. Цикл типа do-while. Требуется угадать заданное в программе число.

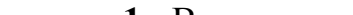
Возможный вариант циклической части программы имеет следующий вид:

```
...
do {cout<<"\nВведите число"; cin >> r;}
while(r!=13);
cout << "Вы успешно угадали число.";
```

В данном примере пользователь вводит числа до тех пор, пока не будет введено число 13. После этого выдается сообщение об успешном угадывании числа.

Вложенные циклы

Вложенным циклом называют конструкцию, в которой один цикл выполняется внутри другого. Внутренний цикл выполняется полностью во время каждой итерации внешнего цикла.

Пример 1. Вложенные циклы. Требуется заполнить весь экран символами '#'.


```
...
int i,k;
for(i=1;i<=25;i++)
    for(k=1;k<=80;k++) cout << '#';
...
```

В данной программе 25 раз осуществляется вывод по 80 символов ‘#’.

В программе можно использовать любые комбинации вложенных циклов: while, for, do-while, если этого требует логика построения программы.

Для того, чтобы корректно выбрать тип используемого цикла, необходимо решить нужен ли цикл с предусловием или с постусловием. Чаще всего необходим цикл с предусловием.

Циклы с предусловием предпочтительнее по двум причинам: первая - чтобы тело цикла выполнялось только при выполнении условия и полностью игнорировалось, если условие не выполняется и вторая -

то, что программа более понятна, когда проверяемое условие находится в начале, а не в конце циклического участка.

В случае выбора цикла с предусловием возникает второй вопрос: что лучше, `for` или `while`?

В принципе все, что можно сделать с помощью одного цикла, можно сделать и с помощью другого. Исходя из соображений правильного стиля программирования, цикл `for` более удобен, когда в цикле используется инициализация и коррекция управляющей переменной. А цикл `while` удобнее применять, когда этого не требуется.

Цикл `for` в языке СИ является более гибкими, за счет трех выражений в скобках после `for`.

Рассмотрим некоторые возможности применения циклов типа `for`.

Пример 1. Счет в порядке убывания.

Требуется вычислить y^5 . Возможное решение имеет вид:

```
...
int r,i,y
for(i=5,r=1;i>=1;i- -) r=r*y;
cout << "ув пятой степени = " << r;
```

Пример 2. Приращение счетчика, отличное от 1.

```
...
int n;
for(n=5;n<61;n+=15) cout << n;
```

Пример 3. Использование символов в качестве счетчика.

Требуется напечатать алфавит. Возможное решение имеет вид:

```
...
char ch;
for(ch='A'; ch<='z';ch++) cout << ch;
```

Пример 4. Проверка условия, отличного от подсчета числа итераций.

Управляющая переменная инициализируется перед циклом.

```
...
int a=2,n;
for(n=3;a<=162;)a=a*n;
```

Здесь n всегда $=3$. а будет меняться 6, 18, 54, 162. Цикл выполнится 4 раза.

Пример 5. Изменение счетчика в геометрической прогрессии.

Требуется подсчитать долг + 10%. Возможное решение имеет вид:

```
...
float k;
for(k=100.0;k<185.0;k*=1.1) cout << "Долг + проценты "<<k;
...
```

Пример 6. Использование выражения в качестве счетчика.

```
...
int k,z;
for(k=1;z<196; z=5*k+++23 )cout << z;
...
```

Пример 7. Неполный список выражений в заголовке цикла.

Можно пропускать одно или несколько выражений. (При этом нельзя пропускать символы «точка с запятой».)

```
...
int p,n,k;
for(p=202; p>=2;) p=p--+n/k;
...
```

Пример 8. Произвольное первое выражение в заголовке цикла.

Первое выражение не обязательно должно инициализировать переменные, оно может быть любым.

```
...
for(cout << "Вводите числа."; p<=30;) cin >> p;
...
```

Пример 9. Изменение шага приращения счетчика в теле цикла.

```
...
float delta=0.1, S=0.0, k;
for (k=1; k<500; k+=delta)
{
    S=S+k*k/150.0;
    if (S>200) delta=0.5;
}
...
```

Пример 10. Использование операции «запятая» в спецификации цикла.

Использование операции «запятая» в спецификации цикла позволяет включать несколько инициализирующих и корректирующих выражений.

```
...
```

```

int i,r,y=2;
for(i=1, r=1;i<=10; i++,r*=y);
cout << "y в степени "<<i-1<<" равно: "r;
...

```

Большая свобода выбора вида выражений, управляющих работой цикла `for`, позволяет с помощью этого оператора делать гораздо больше дополнительных действий, чем выполнять просто фиксированное число итераций.

Нельзя ставить точку с запятой после заголовка цикла. Если поставить точку с запятой, то тело цикла будет состоять из одного пустого оператора «;».

Управляющие операторы в циклах

Кроме условных операторов и операторов циклов существуют еще три оператора, предназначенные для управления порядком выполнения программы: *break*, *continue*, *goto*.

Они применяются реже, поскольку частое их использование ухудшает наглядность программы и увеличивает вероятность ошибок.

Оператор *break* является наиболее важным из этих трех операторов и уже встречался при рассмотрении оператора выбора *switch*.

Он может использоваться в циклах всех трех типов. Выполнение оператора *break* приводит к выходу из цикла, в котором он содержится, и переходу к следующему за циклом оператору.

Если оператор *break* находится внутри вложенных циклов, то его действие распространяется только на тот цикл, непосредственно в котором он находится.

Пример 11. Использование оператора *break*. Требуется угадать задуманное число с 10 попыток.

```

...
int i=1,rez;
while(i++<=10)
{
    cout << "\nУгадайте число";
    cin>>rez;
    if(rez ==15)break;
    cout << "\nПопытка неудачная.";
}
if(i!=12) cout << "\nВы угадали число.";
else cout << "\nВсе попытки исчерпаны";
...

```

При угадывании числа происходит прекращение выполнения цикла с помощью оператора *break*.

Оператор *continue* может использоваться только среди операторов тела цикла. Этот оператор вызывает пропуск оставшейся части итерации внутри цикла и переход к следующей итерации.

Пример 12. Использование оператора *continue*.

Вводятся календарные числа месяца для обработки. Необходимо проверить правильность ввода. Число 31 обозначает конец обработки.

```
...
while(day!=31)
{
    cout << "\nЗадайте календарное число";
    cin >> day;
    if(day<1||day>31) continue;
    {...} // Блок операторов по обработке календарного числа day
}
```

Здесь ввод неправильного значения приводит к пропуску части итерации, предназначенной для обработки этого значения.

Можно изменить условие `if(day<1||day>31)` на обратное `if(day>0 && day<32)`, что позволяет исключить *continue*.

С другой стороны, с помощью *continue* иногда можно сократить некоторые программы, особенно если они включают в себя вложенные операторы `if else`.

Оператор *goto* (переход на заданную метку).

Форма записи:

`goto метка`

...

метка: ОПЕРАТОР;

Например:

`char ch;`

`top: cout << "\nУгадайте символ";cin>> ch;`

`if(ch!='*')`

`goto top;`

`cout << "\n Вы угадали";`

В языке СИ++ оператор *goto* является плохим средством. Его использование значительно усложняет логику программы и идет вразрез с правильным стилем программирования. Существует лишь один случай, когда программисты профессионалы допускают использование *goto*, — это выход из вложенного набора циклов при обнаружении ошибок, так как *break* дает возможность выхода лишь из одного цикла.

Указатели

Каждая ячейка памяти имеет две характеристики – адрес и значение.

В программировании имя переменной это символическое обозначение ее значения. В языке СИ++ кроме имен переменных существуют указатели.

Указатель — это символическое обозначение адреса.

Операции с указателями

В СИ++ имеются две в каком-то смысле обратные друг другу операции:

- операция определения адреса – *&* и
- операция косвенной адресации – ***.

С помощью операции *&* определяется адрес ячейки памяти, содержащей заданную переменную. Например, если *vr* — имя переменной, то *&vr* — адрес этой переменной или *указатель на переменную vr*. Указатель *&vr* — это число, которое является *константой типа указатель*.

В СИ++ также существуют и переменные типа указатель. Значением переменной типа указатель служит адрес переменной или объекта. Пусть переменная типа указатель имеет имя *ptr*, тогда в качестве значения ей можно присвоить адрес переменной *vr* с помощью следующего оператора:

```
ptr=&vr;
```

Противоположная *операция косвенной адресации* *** позволяет обратиться к переменной не напрямую по имени, а косвенно через указатель, содержащий адрес этой переменной.

Данная операция является унарной и выполняется слева направо. Ее не следует путать с бинарной операцией умножения.

Пусть *ptr* — указатель, тогда **ptr* — это значение переменной, на которую указывает *ptr*.

Тогда:

```
a=b;      // присвоит переменной a значение переменной b
```

```
ptr=&b;    // присвоит переменной a значение переменной b через
```

указатель

```
a=*ptr;
```

Объявление и использование указателей

Объявление указателей выполняется с помощью операторов вида:

```
<тип переменной> *<имя указателя на переменную заданного  
типа>;
```

Пример 13. Объявления указателей.

```
int *ptri;           //указатель на переменную целого типа
char *ptrc;          //указатель на переменную символьного типа
float *ptrf;         //указатель на переменную с плавающей точкой
```

Такой способ объявления указателей возник из-за того, что переменные разных типов занимают различное число ячеек памяти.

Пример 14. Использование указателей.

```
int a=10, b=5;
int *ptr=&a; //инициализация указателя адресом переменной a
cout << "Адрес переменной a = "<< ptr << "\n Значение переменной a
="<<*ptr;
ptr=&b; //теперь ptr указывает на переменную b
cout << "Адрес переменной b = "<< ptr << "\n Значение переменной b
="<<*ptr;
```

Отметим, что без указателей невозможна обработка объектов. Например, передать объект некоторой функции можно только через указатель.

Литература

1. Ф. М. Яблонский, Ю. В. Троицкий. Средства отображения информации. М.; Высшая школа, 1985-200с.
2. Быстров Ю. А., Литвак И. И., Першанов Г. М. Электронные приборы для отображения информации. М.; Радио и связь, 1985-162с.
3. Телевидение / Под ред. Проф. П. В. Шмакова. – М.: Связь, 1979-432 с.
4. Хоган Т. Аппаратные и программные средства персональных компьютеров. Справочник в двух томах. – М.: Радио и связь. – 1995. – Том 2. – 260 с.
5. Мягеев А. А., Степанов В. Н., Персональные ЭВМ и микро-ЭВМ. – Радио и связь, 1991. – 171 с.

СОЛДАТОВ Алексей Иванович
СОРОКИН Павел Владимирович
СОЛДАТОВ Андрей Алексеевич
КИСЕЛЕВА Екатерина Юрьевна

ИССЛЕДОВАНИЕ ЭЛЕКТРОННЫХ ИНДИКАТОРОВ. ПРАКТИКУМ

Учебно-методическое пособие

Редактор

Подписано к печати

Формат 60х84/16. Бумага офсетная.

Печать RISO. Усл. печ. л. . Уч.-изд. л. .

Тираж экз. Заказ . Цена С.

Издательство ТПУ. 634050, Томск, пр. Ленина, 30.