Динамическая индикация на базе светодиодных матриц в программной среде Proteus

Татьяна Колесникова (beluikluk@gmail.com)

В статье рассмотрены возможности программы Proteus по реализации динамической индикации на базе точечных светодиодных матриц разрешением 8×8 и 16×16. Кроме того, приведены примеры моделирования схем вывода информации с использованием микросхем матриц, работающих под управлением микроконтроллера ATmega16, компиляция программы инициализации которого выполнена в CodeVisionAVR.

Введение

Светодиодная матрица – это графический индикатор, который можно использовать для вывода простых изображений, букв и цифр, математических и специальных знаков, символов. Матричные светодиодные индикаторы позволяют отображать статический и анимированный текст (все символы латинского и русского алфавитов в достаточно узнаваемом виде) и графику и применяются в устройствах отображения информации различного характера, например в информационных табло в общественных местах (супермаркетах, банковских учреждениях и т.п.).

Для проектирования устройства вывода информации и моделирования его работы удобно использовать программную среду Proteus (в данном случае версии 8.1), библиотека компонентов которой содержит как аналоговые, так и цифровые компоненты, а также светодиодные матрицы и микроконтроллеры с возможностью их программирования.

При проектировании устройства вывода информации, работающего под управлением микроконтроллера AVR, написание программы инициализации и её компиляцию можно выполнить с помощью CodeVisionAVR 3.12 (интегрированной среды разработки программного обеспечения для микроконтроллеров семейства AVR фирмы Atmel, которая имеет в своём составе компилятор языка C для AVR). CodeVisionAVR поддерживает все базовые конструкции языка C, которые используются при написании программ (алфавит, константы, идентификаторы, комментарии) и разрешены архитектурой AVR, с некоторыми добавленными характеристиками, реализующими преимущество специфики архитектуры AVR. Используя специальные директивы, в любом месте программы можно включить ассемблерный код. Программные средства позволяют напрямую обращаться к регистрам микроконтроллера, управлять состоянием линий портов, а следовательно, и состоянием светодиодов (зажжён/отключён) проектируемого на базе светодиодных матриц устройства.

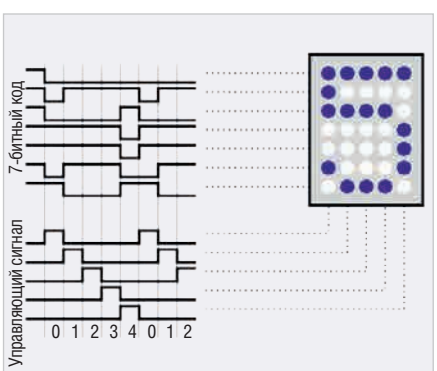

Рис. 1. Принцип работы точечного индикатора разрешением 5×7 пикселей

Матричный индикатор представляет собой массив светодиодов, объединённых в один корпус, и позволяет управлять каждым из них. Множество светодиодов сгруппировано в столбцы и строки. Разрешение матричного индикатора – это количество светодиодов по горизонтали и вертикали. Самые распространённые индикаторы имеют разрешение 8×8 точек (светодиодов). Встречаются матрицы разрешением 5×7, 5×8, 16×16. Если требуется светодиодная матрица большого разрешения, то её составляют из нескольких индикаторов.

Для управления светодиодной матрицей применяются два распространённых метода: статический и динамический. Для упрощения схемы управления, а также для сокращения количества выводов индикатора используется динамический способ управления, который подразумевает поочерёдное включение различных групп элементов отображения со скоростью, превышающей время реакции человеческого глаза. Несмотря на то что изображение на индикаторе при таком способе управления в каждый момент времени неполное, глаз человека интегрирует его и видит целостную картинку. Динамическая индикация широко применяется для отображения различной информации, например температуры, напряжения, времени или просто количества срабатываний каких-либо устройств или датчиков.

Для реализации динамического способа управления все светодиоды в столбцах объединяются по анодам, а в рядах (строках) – по катодам (существуют варианты исполнения матричных индикаторов, в которых светодиоды в рядах объединяются по анодам). На аноды последовательно подаются положительные периоды напряжения, на катоды подаётся отрицательный сигнал. Чтобы сформировать полное изображение, необходимо для каждого столбца последовательно установить соответствующий код. Так, в случае индикатора на 5 столбцов по 7 точек в столбце необходимо установить для каждого из столбцов код из 7 бит. При этом чтобы зажечь точки, расположенные в нескольких строках определённого столбца светодиодной матрицы, нужно подать на вывод этого столбца логическую единицу, а на выводы этих строк – логический ноль (если в индикаторе светодиоды в столбцах объединены по анодам). В результате этого зажгутся светодиоды, которые находятся на пересечении столбца и строк.

Принцип работы точечного индикатора разрешением 5×7 демонстрирует рисунок 1, где светодиоды в строках объединены по катодам, а в столбцах по анодам. Чтобы сформировать полное изображение, для каждого столбца последовательно подаётся свой 7-битный код.

Проектирование принципиальной электрической схемы в Proteus

В Proteus светодиодные матрицы собраны в разделе *Dot Matrix Displays* библиотеки *Optoelectronics* и представлены точечными индикаторами разрешением 5×7 и 8×8 со светодиодами синего, зелёного, оранжевого и красного цветов. Также в программе есть

возможность создания и добавления в библиотеку компонентов пользовательских моделей, например матрицы разрешением 16×16 точек. Управление подсвечиванием точек выполняется путём подачи комбинаций логических нулей и единиц на выводы матрицы. Верхние выводы соответствуют строкам, а нижние – столбцам таблицы светодиодов.

К примеру, если необходимо зажечь два светодиода в 3-й и 4-й строках 2-го столбца (см. рис. 2а) точечного индикатора разрешением 8×8, то нужно на 2-й нижний вывод подать логическую единицу, а на 3-й и 4-й верхние выводы – логический ноль. Для того чтобы зажечь пять последних точек в 7-й и 8-й строках матрицы (см. рис. 2б), нужно подать логическую единицу на 4–8-й нижние выводы и логический ноль на 7–8-й верхние выводы. На свободные верхние выводы подаётся логическая единица, на нижние выводы – логический ноль.

Аналогичным образом управляют светодиодами матрицы разрешением 16×6. Например, чтобы зажечь шесть светодиодов в 6–11-й строках 7–11-го и 15-го столбца, нужно подать логическую единицу на 7–11-й нижние выводы и логический ноль на 6–11-й верхние выводы (см. рис. 2в).

В представленном примере значения логического нуля и единицы подаются на выводы микросхем точечных матриц с помощью инструментов *GROUND и POWER* панели *TERMINALS*. Открывают панель нажатием кнопки *Terminals Mode* на левой панели инструментов редактора *ISIS*.

Для управления матричным индикатором можно использовать микроконтроллер с достаточным количеством линий

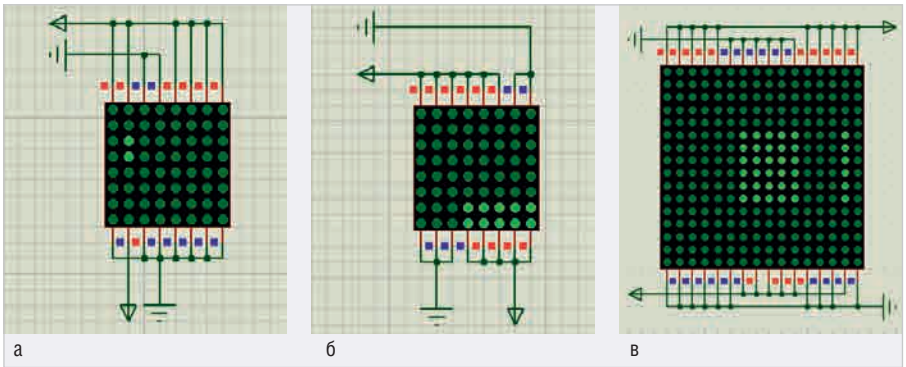

Рис. 2. Подсветка в матрицах разрешением 8×8 и 16×16: а) двух светодиодов во втором столбце; б) группы светодиодов в правом нижнем углу матрицы; в) группы светодиодов в 7–11-м и 15-м столбцах

ввода/вывода. Каких-либо стандартных правил сопряжения микроконтроллеров с точечными матрицами не существует. На практике для ограничения величины тока, протекающего через светодиоды, между выводами портов микроконтроллера и анодов светодиодов индикатора включают резисторы номиналом 300 Ом.

Создадим в Proteus новый проект и добавим в рабочее поле микросхему светодиодной матрицы зелёного цвета разрешением 8×8, для чего с помощью команды контекстного меню *Place* → *Component* → *From Libraries* редактора *ISIS* откроем окно *Pick Devices* и выберем левой кнопкой мыши из раздела *Dot Matrix Displays* библиотеки *Optoelectronics* микросхему MATRIX-8X8-GREEN (см. рис. 3а). Нажмём кнопку *OK* (окно *Pick Devices* будет закрыто) и разместим микросхему в рабочей области проекта.

Для тестирования работы светодиодной матрицы будем использовать микроконтроллер, в качестве которого применим микросхему ATmega16, представленную в разделе *AVR Family* библиотеки *Microprocessor ICs* (см. рис. 3б).

Подсоединим верхние выводы микросхемы светодиодной матрицы к линиям PC0–PC7, а нижние к линиям PD0–PD7 микроконтроллера ATmega16, как показано на рисунке 4. В данном примере управление строками светодиодной матрицы осуществляется с помощью выводов порта PC микроконтроллера, управление столбцами – с помощью выводов порта PD. Выбор линий портов микроконтроллера для подключения к выводам светодиодной матрицы производится разработчиком произвольно.

Создание программного кода в CodeVisionAVR

Формирование программного кода в CodeVisionAVR выполняют при помощи автоматического генератора *CodeWizardAVR* или вручную с нуля, используя синтаксис языка программирования С и функции стандартных библиотек программы. Удобство применения генератора состоит в быстром получении кода выполнения функций

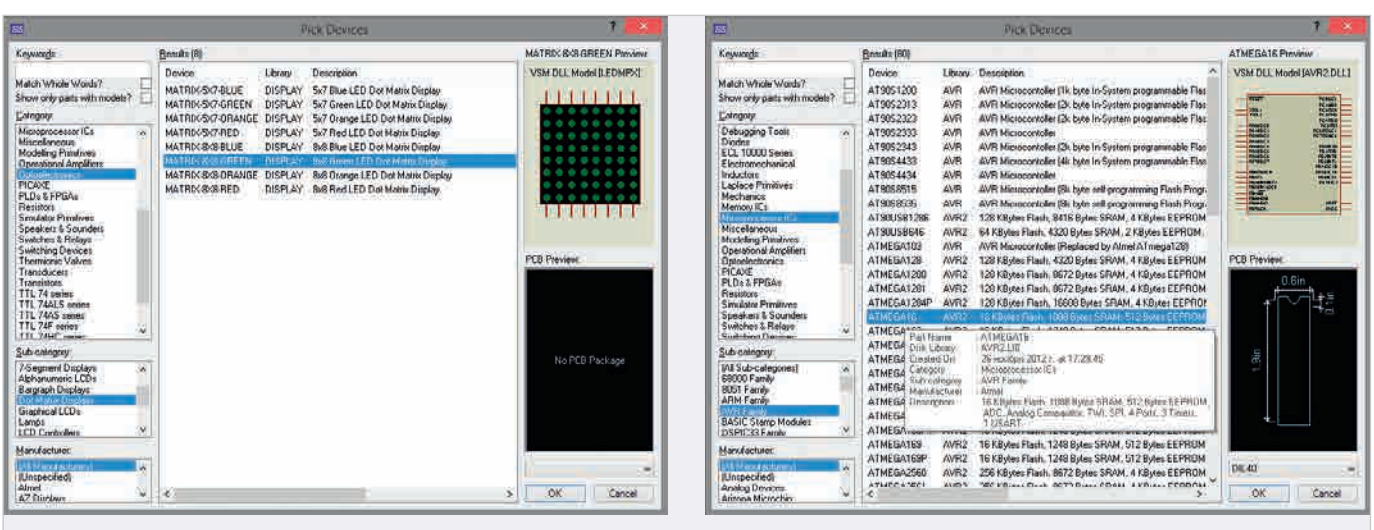

Рис. 3. Окно *Pick Devices* редактора *ISIS*: а) раздел *Dot Matrix Displays* библиотеки *Optoelectronics*; б) раздел *AVR Family* библиотеки *Microprocessor ICs*

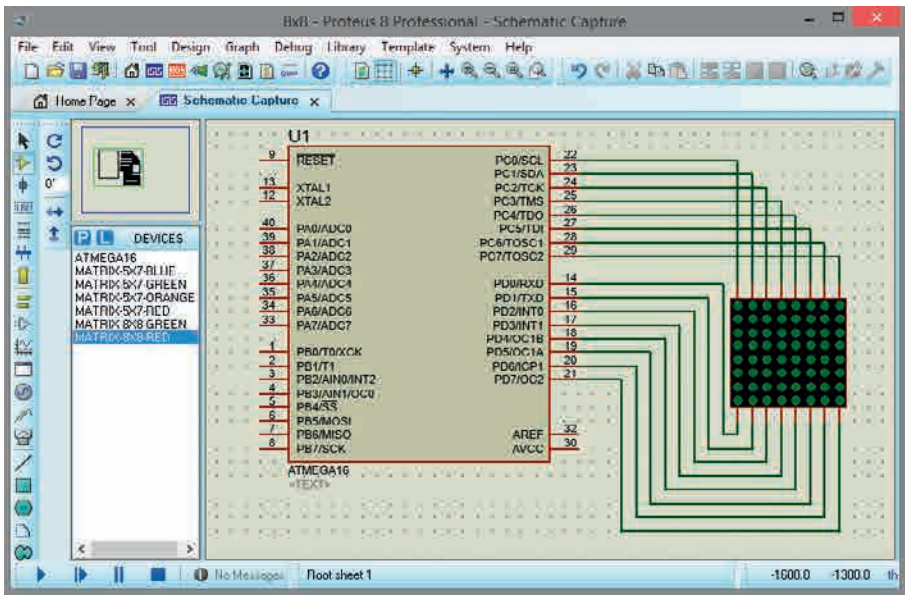

Рис. 4. Сопряжение микроконтроллера ATmega16 и микросхемы светодиодной матрицы разрешением 8×8

инициализации микроконтроллера и его портов ввода/вывода, аналогового компаратора, таймеров/счётчиков, интерфейсов UART и SPI, буквенно-цифровых и графических дисплеев и др. Однако в процессе работы мастера формируется достаточно объёмный код, который впоследствии приходится редактировать.

После создания с помощью команды основного меню *File* → *New* → *Project* нового проекта в CodeVisionAVR открывается окно генератора кода *CodeWizardAVR*, где задаются параметры микроконтроллера, его внутренних ресурсов и используемых в схеме периферийных устройств. В данном примере это тип и частота работы микроконтроллера (вкладка *Chip Settings* – см. рис. 5а), опции портов ввода/вывода микроконтроллера (вкладка *Ports Settings* – см. рис. 5б). Важно, чтобы значение тактовой частоты микроконтроллера, указанное в поле *Clock* вкладки *Chip Settings*, совпадало со значением в поле *CKSEL Fuses* его окна свойств в Proteus (в данном примере это 8 МГц). Окно генератора кода в *CodeVisionAVR* можно также открыть нажатием пиктограммы *Run the CodeWizardAVR* панели *Tools*, которую добавляют в проект с помощью команды *View* → *Toolbars* → *Tools* основного меню.

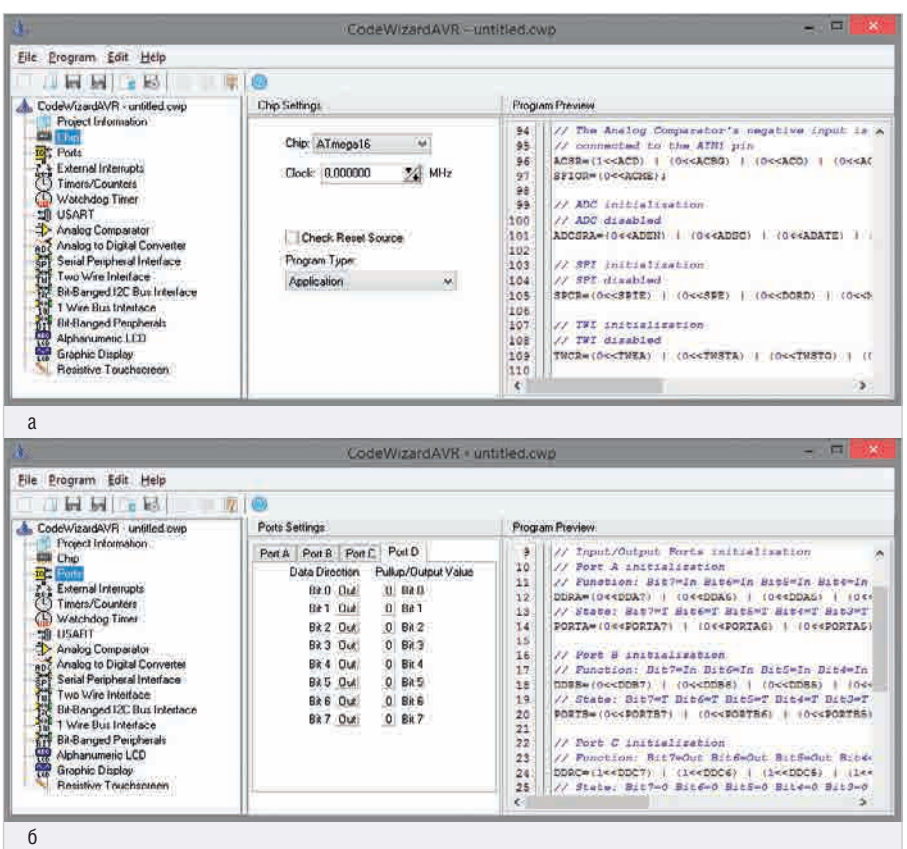

Рис. 5. Окно *CodeWizardAVR*: а) настройка параметров микроконтроллера; б) настройка параметров портов ввода/вывода микроконтроллера

На вкладке *Ports Settings* для каждого отдельного порта микроконтроллера отведена своя закладка, где в поле *Direction* щелчком левой кнопки мыши можно выбрать одно из значений битов порта: *Out* (линия порта работает на вывод данных), *In* (линия порта работает на приём данных). В данном примере для бит Bit 0 – Bit 7 портов Port D и Port C нужно указать значение *Out*.

Предварительный просмотр кода программы, который генерируют с помощью команды *Program* → *Generate* основного меню, выполняют в поле *Program Preview*. После настройки параметров и генерации кода с помощью команды *Program* → *Generate*, *Save and Exit* основного меню или пиктограммы *Generate program*, *save and exit* верхней панели инструментов окно *CodeWizardAVR* будет автоматически закрыто. Полученный код отобразится в окне кода CodeVisionAVR, где и будет вестись дальнейшее написание программы. При этом автоматически сгенерированный код можно отредактировать на своё усмотрение.

Записывая в порт PD соответствующие биты, микроконтроллер может переключать столбцы матрицы. Между переключениями столбцов микроконтроллер должен выдержать паузу. Чтобы зажечь нужные светодиоды в столбце, необходимо программным путём записать в порт PC микроконтроллера соответствующий двоичный код. К примеру, чтобы зажечь три первых и один последний светодиоды второго столбца матричного индикатора разрешением 8×8, необходимо в порт PC записать двоичный код 0b01111000, а в порт PD – код 0b00000010. При таком способе подачи управляющих сигналов будут подсвечены сразу все светодиоды столбца, на катоды которых был подан логический ноль. Чтобы получить полную картинку, вывод столбцов символа/рисунка выполняется последовательно в цикле.

Для уменьшения мерцания отображаемых на индикаторе символов подсвет-

Таблица 1. Двоичный код символа «5»

1	1	0	0	0	0	0	1	1
2	1	0	0	1	1	1	1	1
3	1	0	0	0	0	0	1	1
4	1	0	0	1	1	0	0	1
5	1	1	1	1	1	0	0	1
6	1	1	1	1	1	0	0	1
7	1	0	0	1	1	0	0	1
8	1	1	0	0	0	0	1	1
В.в.м. / Н.в.м.	1	2	3	4	5	6	7	8

Примечание: В.в.м. – верхние выводы матрицы светодиодов; Н.в.м. – нижние выводы матрицы светодиодов.

ку светодиодов в столбце можно выполнять по одному последовательно. Тогда в программе инициализации микроконтроллера вместо одной команды PORTC=0b01111000, с помощью которой подаются управляющие сигналы (логические нули) на катоды матричного индикатора, нужно записать четыре команды PORTC=0b01111111, PORTC=0b11111110,PORTC=0b11111101, PORTC=0b11111011 и, соответственно, четыре раза выдержать паузу длительностью в 1 мс – команда `delay_ms (1)`.

Вывод символов на матричном индикаторе разрешением 8×8

В качестве примера рассмотрим вывод на экран микросхемы точечного индикатора разрешением 8×8 цифры 5. Двоичный код данного символа представлен в таблице 1. Точки, которые нужно зажечь на матричном индикаторе, обозначены в таблице нулями, точки, которые не используются – единицами.

Формирование изображения на индикаторе выполним при помощи программы, написанной на языке С с включением с помощью директив `#asm` и `#endasm` ассемблерного кода, что позволит использовать регистры R0–R31 микроконтроллера. При этом к регистру данных порта PD можно обращаться по адресу ввода/вывода 0x12, а к регистру данных порта PC – по адресу 0x15.

Текст программы представлен в листинге 1.

Введём текст программы в окне кода CodeVisionAVR и с помощью команды основного меню *Project* → *Build All* запустим компиляцию, по окончании которой выдаётся отчёт о наличии ошибок в коде программы. Если ошибки не обнаружены, на диске компьютера будет создан hex-файл для записи в микроконтроллер.

Теперь перейдём в Proteus и в окне свойств микросхемы ATmega16 (окно открывают двойным щелчком левой кнопки мыши по выбранному на схеме микроконтроллеру) в поле *Program File* укажем путь к файлу прошивки на диске компьютера (см. рис. 6). С помощью команды основного меню *Debug* → *Run Simulation* запустим в Proteus моделирование собранной схемы (результат представлен на рисунке 7) и проанализируем её работу.

Программным путём были даны указания микроконтроллеру через порт PD отправить на нижние выводы микросхемы точечного индикатора двоич-

Листинг 1

```
#include <mega16.h> // подключение заголовочных файлов
#include <delay.h>

void main(void)
{
PORTD=PORTC=0x00; // инициализация портов
DDRD=0xff; // порт PD работает на вывод данных
DDRC=0xff; // порт PC работает на вывод данных
#asm
ldi R19, 20 ; // запись в регистр R19 числа 20

Loop: ; // начало цикла вывода рисунка на экран дисплея
clr R18 ; // очистка регистра R18
clr R17 ; // очистка регистра R17
ldi R18, 0b00000010
out 0x12, R18 ; // активизация второго столбца точечной матрицы
ldi R17, 0b10110000
out 0x15, R17 ; // вывод первого столбца символа «5»
rcall pause ; // вызов подпрограммы задержки

clr R18 ; // очистка регистра R18
clr R17 ; // очистка регистра R17
ldi R18, 0b00000100
out 0x12, R18 ; // активизация 3-го столбца точечной матрицы
ldi R17, 0b00110000
out 0x15, R17 ; // вывод второго столбца символа «5»
rcall pause ; // вызов подпрограммы задержки

clr R18 ; // очистка регистра R18
clr R17 ; // очистка регистра R17
ldi R18, 0b00011000
out 0x12, R18 ; // активизация 4-го и 5-го столбцов точечной матрицы
ldi R17, 0b01111010
out 0x15, R17 ; // вывод 3-го и 4-го столбцов символа «5»
rcall pause ; // вызов подпрограммы задержки

clr R18 ; // очистка регистра R18
clr R17 ; // очистка регистра R17
ldi R18, 0b00100000
out 0x12, R18 ; // активизация 6-го столбца точечной матрицы
ldi R17, 0b00000010
out 0x15, R17 ; // вывод 5-го столбца символа «5»
rcall pause ; // вызов подпрограммы задержки

clr R18 ; // очистка регистра R18
clr R17 ; // очистка регистра R17
ldi R18, 0b01000000
out 0x12, R18 ; // активизация 7-го столбца точечной матрицы
ldi R17, 0b10000111
out 0x15, R17 ; // вывод 6-го столбца символа «5»
rcall pause ; // вызов подпрограммы задержки
rjmp Loop ; // бесконечный цикл

pause: ; // подпрограмма задержки
subi  R19,1 ; // вычесть 1 из значения регистра R19
brne  pause ; // перейти на начало подпрограммы, если значение регистра R19 не равно 1
ldi R19, 20 ; // если значение регистра R19 равно 1, записать в регистр число 20
ret ; // выход из подпрограммы
#endasm
}
```

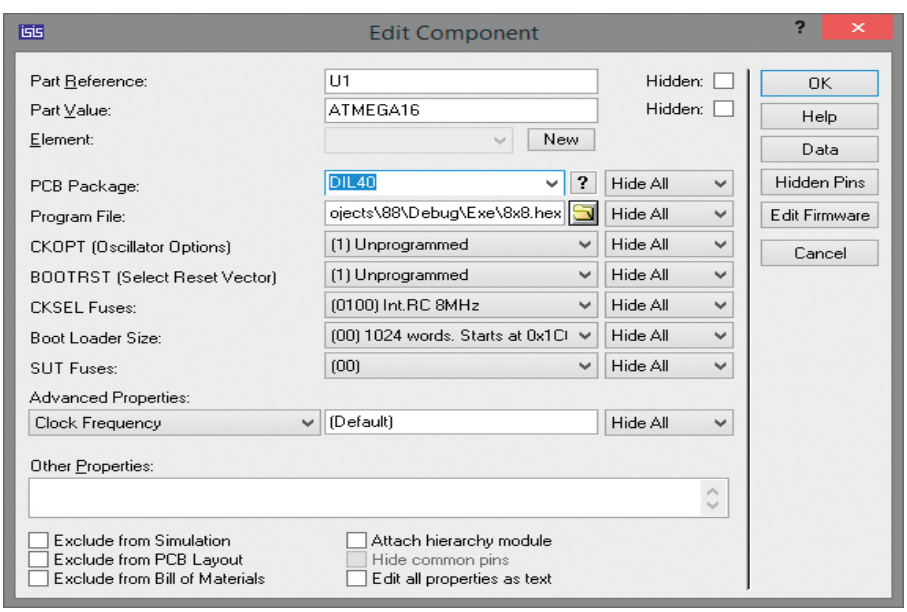

Рис. 6. Окно настроек микроконтроллера ATmega16

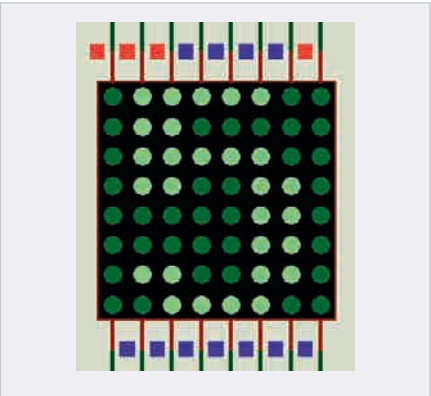

Рис. 7. Результат работы программы вывода символа на дисплей точечной светодиодной матрицы разрешением 8×8

ный код 0b00000010, который активизирует второй столбец матрицы. При этом через порт PC на верхние выводы микросхемы точечного индикатора подаётся двоичный код 0b10110000, который включает 5 светодиодов второго столбца матрицы (см. рис. 8а). После паузы длительностью 20 мс в порт PD программным путём подаётся двоичный код 0b00000100, который активизирует третий столбец матрицы. В порт PC подаётся двоичный код 0b00110000, который включает 6 светодиодов в этом столбце (см. рис. 8б). После паузы длительностью 20 мс в порт PD подаётся двоичный код 0b00011000, который активизирует два столбца матрицы (4-й и 5-й). Как в 4-м, так и в 5-м столбцах точечной матрицы нужно включить светодиоды, расположенные в 1-й, 3-й и 8-й строках (см. рис. 8в), поэтому в данном случае удобно активизировать сразу два столбца и подать через порт PC микроконтроллера на верхние выводы индикатора двоичный код 0b01111010, значения 1-го, 3-го и 8-го бита которого равны нулю. После паузы длительностью 20 мс в порт PD подаётся двоичный код 0b00100000, активизирующий 6-й столбец матрицы, в котором в результате записи кода 0b00000010 в порт PC будут подсвечены 1-й и с 3-го по 8-й светодиоды (см. рис. 8г). Далее после паузы программа инициализации микроконтроллера выводит в порт PD двоичный код 0b01000000, который активизирует 7-й столбец матрицы. При этом через порт PC на верхние выводы микросхемы точечного индикатора подаётся двоичный код 0b10000111, который включает четыре светодиода (см. рис. 8д).

Когда на дисплее отрисовано всё изображение, последовательность действий повторяется сначала. В результате на дисплее точечного индикатора будет отображён символ (цифра 5). Статическое изображение получено вследствие непрерывного выполнения в цикле последовательной активизации заданных столбцов точечной матрицы и подсветки в них указанных светодиодов.

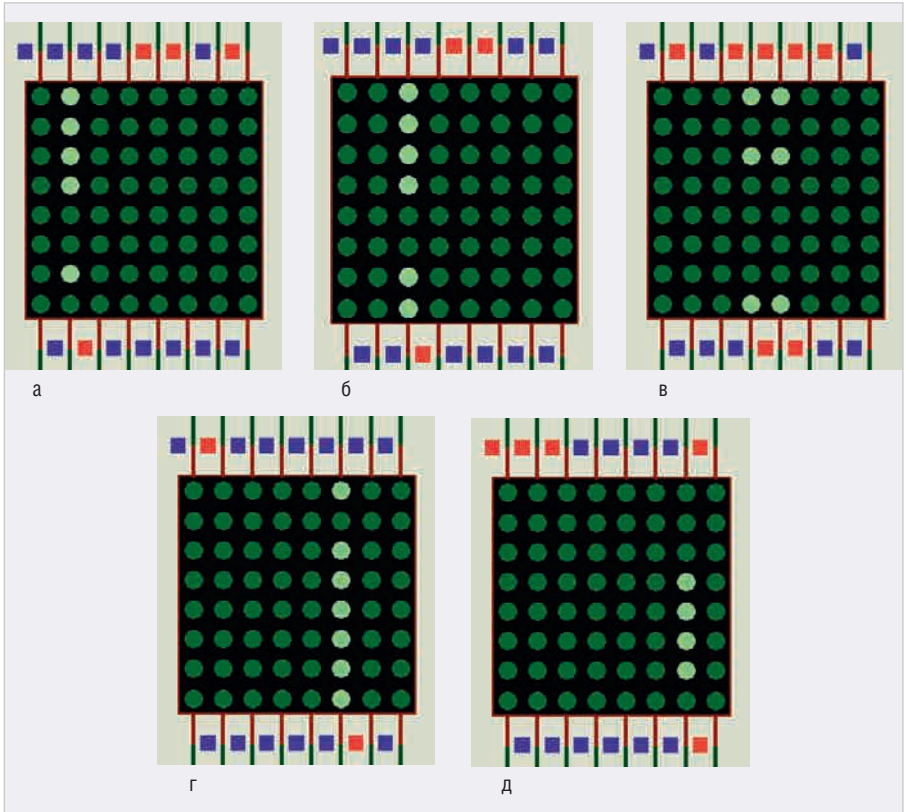

Рис. 8. Процесс формирования изображения на дисплее точечной светодиодной матрицы разрешением 8×8: а) вывод первого столбца символа; б) вывод второго столбца символа; в) вывод 3-го и 4-го столбцов символа; г) вывод 5-го столбца символа; д) вывод 6-го столбца символа

Листинг 2

```
while(1) { // начало цикла
PORTC=0b10110000; // вывод первого столбца символа «5»
PORTD=0b00000010;
delay_ms(20); // задержка 20 мс
PORTC=0b00110000; // вывод второго столбца символа «5»
PORTD=0b00000100;
delay_ms(20);
PORTC=0b01111010; // вывод 3-го и 4-го столбца символа «5»
PORTD=0b00011000;
delay_ms(20);
PORTC=0b00000010; // вывод 5-го столбца символа «5»
PORTD=0b00100000;
delay_ms(20);
PORTC=0b10000111; // вывод 6-го столбца символа «5»
PORTD=0b01000000;
delay_ms(20);

}
```

Для формирования изображения на индикаторе можно применить и код, написанный на языке C (см. листинг 2), однако в результате применения подпрограммы, написанной на ассемблере, на дисплее точечной матрицы была получена более чёткая и стабильная картинка.

Рассмотрим формирование изображения с эффектом бегущей строки. Для этого внесём изменения в программу вывода символа (цифры 5), написанную на языке C (см. листинг 3).

Запустим в CodeVisionAVR компиляцию кода программы. По её окончании на диске компьютера будет создан hex-файл, путь к которому укажем в окне свойств микросхемы ATmega16 в Proteus. В поле CKSEL Fuses этого окна установим значение тактовой частоты микроконтроллера 1MHz.

Запустим в Proteus моделирование собранной схемы (см. рис. 9) и проанализируем её работу. Программным путём были даны указания микроконтроллеру через порт PD отправить поочерёдно на нижние выводы микросхемы точечного индикатора двоичный код, который активизирует 2–7-й столбцы матрицы. При этом через порт PC на верхние выводы микросхемы точечного индикатора подаётся двоичный код, который включает нужные светодиоды активного столбца матрицы. Подача зна-

чений выполняется в цикле `for`, в первом проходе которого картинка выводится на дисплей матрицы начиная со второго столбца. Далее при помощи операции `<<1` производится сдвиг столбцов на одну позицию вправо. При этом значения строк остаются неизменными. Таким образом, во втором проходе картинка будет выводиться начиная с третьего столбца. Аналогичным образом выполняются следующие шесть проходов, после чего повторяется цикл `while(1)`. В 7-м проходе будут подсвечены 5 светодиодов в 8-м столбце. В результате на дисплее точечной матрицы будет непрерывно отображаться картинка, последовательно сдвигающаяся вправо.

Часовые диаграммы работы схемы управления светодиодной матрицей, полученные с помощью логического анализатора, представлены на рисунке 10. Сигналы с выводов PC0–PC7 микроконтроллера отображены на графическом экране анализатора зелёным цветом, сигналы, полученные с выводов PD0–PD7 – синим цветом. Подключение логического анализатора к схеме управления светодиодной матрицей показано на рисунке 9.

Последовательный вывод символов на матричном индикаторе разрешением 16×16

Соберём в рабочей области проекта схему с использованием микроконтроллера ATmega16 и матричного индикатора зелёного цвета разрешением 16×16. Подсоединим верхние выводы микросхемы светодиодной матрицы к линиям PA0–PA6, а нижние к линиям PB0–PB4 микроконтроллера ATmega16, как показано на рисунке 11.

Листинг 3

```
#include <mega16.h> // подключение заголовочных файлов
#include <delay.h>
unsigned short column2, column3, column45, column6, column7, num;
void main(void)
{
PORTD=PORTC=0x00; // инициализация портов
DDRD=0xff; // порт PD работает на вывод данных
DDRC=0xff; // порт PC работает на вывод данных
 while(1) { // начало цикла
// начальные значения переменных
column2 = 0b00000010; // активизация 2-го столбца точечной матрицы
column3 = 0b00000100; // активизация 3-го столбца
column45 = 0b00011000; // активизация 4-го и 5-го столбцов
column6 = 0b00100000; // активизация 6-го столбца
column7 = 0b01000000; // активизация 7-го столбца
for (num=0; num<8; num++) { // перебор столбцов матрицы от 1-го до 8-го
PORTC=0b10110000; // вывод первого столбца символа «5»
PORTD = column2; // запись значения переменной column2 в порт PD
delay_ms(20); // задержка
PORTC=0b00110000; // вывод второго столбца символа «5»
PORTD = column3; // запись значения переменной column3 в порт PD
delay_ms(20); // задержка
PORTC=0b01111010; // вывод 3-го и 4-го столбцов символа «5»
PORTD = column45; // запись значения переменной column45 в порт PD
delay_ms(20);
PORTC=0b00000010; // вывод 5-го столбца символа «5»
PORTD = column6; // запись значения переменной column6 в порт PD
delay_ms(20);
PORTC=0b10000111; // вывод 6-го столбца символа «5» на дисплей индикатора
PORTD = column7; // запись значения переменной column7 в порт PD
delay_ms(20);
column2 = column2<<1; // сдвиг первого столбца символа «5»
column3 = column3<<1; // сдвиг второго столбца символа «5»
column45 = column45<<1; // сдвиг третьего и четвертого столбца символа «5»
column6 = column6<<1; // сдвиг пятого столбца символа «5»
column7 = column7<<1; // сдвиг шестого столбца символа «5»
}}}
```

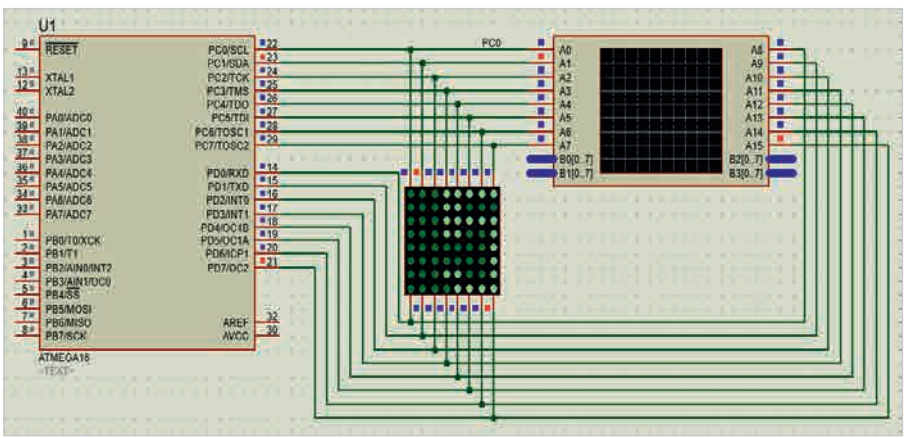

Рис. 9. Формирование изображения с эффектом бегущей строки на дисплее светодиодной матрицы разрешением 8×8

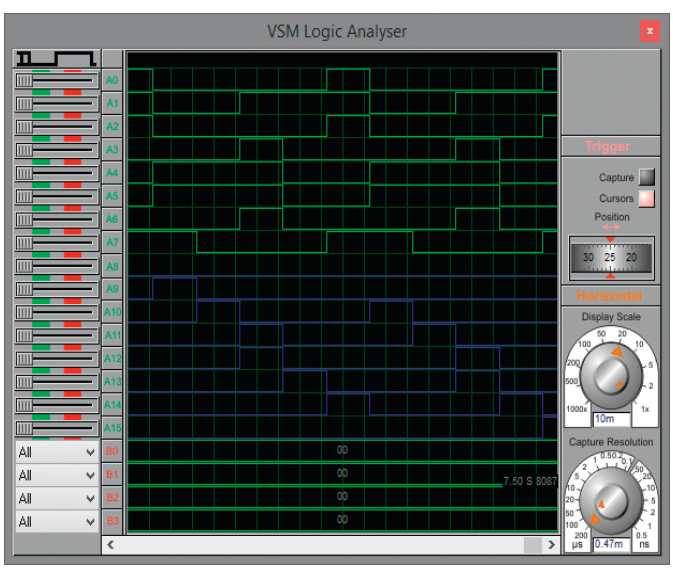

Рис. 10. Часовые диаграммы работы схемы управления светодиодной матрицей разрешением 8×8

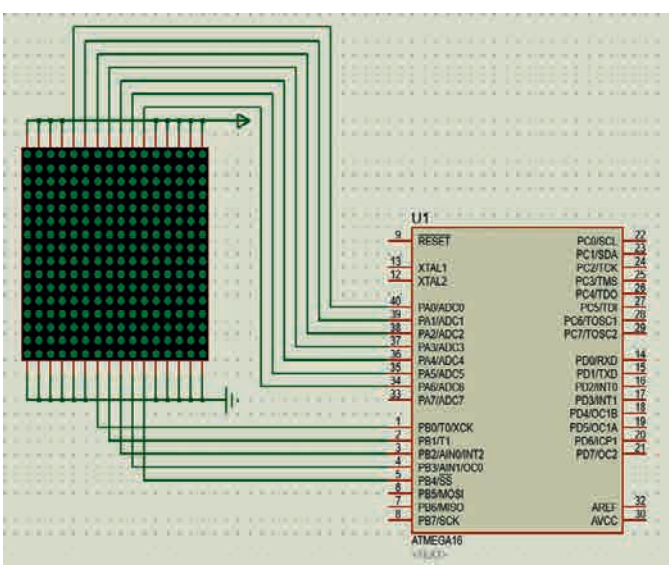

Рис. 11. Сопряжение микроконтроллера ATmega16 и микросхемы светодиодной матрицы разрешением 16×16

Таблица 2. Двоичные коды символов «a», «b», «c»

7	1	1	1	1	1	0	1	Двоичный код символа «a»	0	0	0	0	0	0	0	Двоичный код символа «b»	1	1	1	0	0	0	1	Двоичный код символа «c»
8	1	1	0	1	0	0	0		0	0	0	0	0	0	0		1	1	0	0	0	0	0	
9	1	1	0	0	0	1	0		1	1	0	1	1	1	0		1	1	0	1	1	1	0	
10	1	1	0	0	0	0	0		1	1	0	0	0	0	0		1	1	0	0	1	0	0	
11	1	1	1	0	0	0	0		1	1	1	0	0	0	1		1	1	1	0	1	0	1	
№ н.в / № в.в	5	6	7	8	9	10	11		5	6	7	8	9	10	11		5	6	7	8	9	10	11	

Примечание: № н.в. – номер нижнего вывода матрицы светодиодов; № в.в. – номер верхнего вывода матрицы светодиодов.

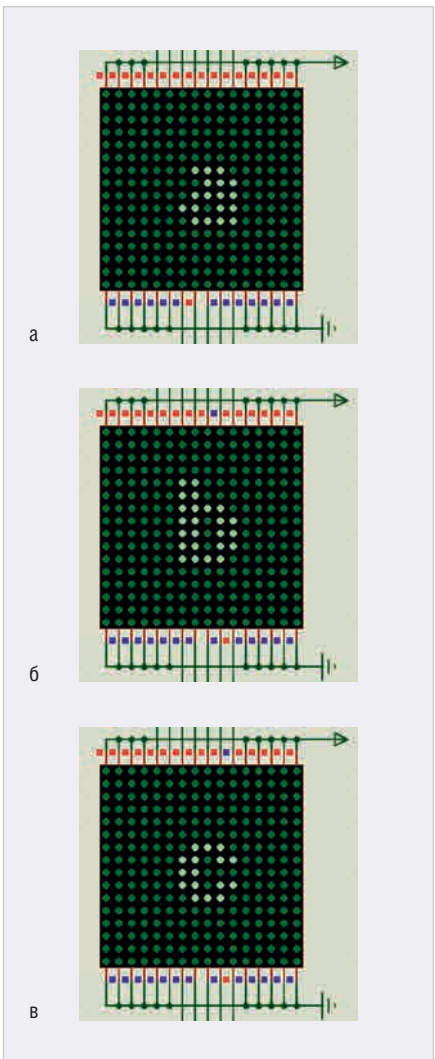

Рис. 12. Результат вывода последовательно сменяющихся символов на дисплей точечной светодиодной матрицы разрешением 16×16:
а) символ «a»; б) символ «b»; в) символ «c»

Листинг 4

```
#include <mega16.h>
#include <delay.h>

const unsigned short a[35]={ // двоичный код символа «a»
0b11111111, 0b11111111, 0b11111111, 0b11111111, 0b11111111,
0b11011111, 0b11111111,
0b11111111, 0b11111111, 0b11111011, 0b11111111, 0b11101111,
0b11011111, 0b10111111,
0b11111111, 0b11111111, 0b11111011, 0b11110111, 0b11101111,
0b11111111, 0b10111111,
0b11111111, 0b11111111, 0b11111011, 0b11110111, 0b11101111,
0b11011111, 0b10111111,
0b11111111, 0b11111111, 0b11111111, 0b11110111, 0b11101111,
0b11011111, 0b10111111,
};

const unsigned short b[35]={ // двоичный код символа «b»
0b11111110, 0b11111101, 0b11111011, 0b11110111, 0b11101111,
0b11011111, 0b10111111,
0b11111110, 0b11111101, 0b11111011, 0b11110111, 0b11101111,
0b11011111, 0b10111111,
0b11111111, 0b11111111, 0b11111011, 0b11111111, 0b11111111,
0b11111111, 0b10111111,
0b11111111, 0b11111111, 0b11111011, 0b11110111, 0b11101111,
0b11011111, 0b10111111,
0b11111111, 0b11111111, 0b11111111, 0b11110111, 0b11101111,
0b11011111, 0b11111111,
};

const unsigned short c[35]={ // двоичный код символа «c»
0b11111111, 0b11111111, 0b11111111, 0b11110111, 0b11101111,
0b11011111, 0b11111111,
0b11111111, 0b11111111, 0b11111011, 0b11110111, 0b11101111,
0b11011111, 0b10111111,
0b11111111, 0b11111111, 0b11111011, 0b11111111, 0b11111111,
0b11111111, 0b10111111,
0b11111111, 0b11111111, 0b11111011, 0b11110111, 0b11111111,
0b11011111, 0b10111111,
0b11111111, 0b11111111, 0b11111111, 0b11110111, 0b11111111,
0b11011111, 0b11111111,
};

unsigned short count, column, num, repeat; // определение типа данных
переменных

void main(void) {
PORTA=PORTB=0x00; // инициализация портов
DDRA=0xff; // порт PA работает на вывод данных
DDRB=0xff; // порт PB работает на вывод данных

do{
for (repeat=0; repeat<25; repeat++){ // вывод символа «a»
column = 0b00000001; // начиная с первого управляемого столбца
for (num=0; num<5; num++) { // выполнение 5 проходов по столбцам
for(count = num*7;count < (num*7+7);count++){ // выполнение прохода в
каждом столбце по точкам матрицы
PORTA = a[count]; // определение состояния каждого светодиода в
столбце

PORTB = column; // запись кода активного столбца в порт PB
delay_ms(1);} // пауза длительностью 1 мс
column = column<<1;} // переход на следующий столбец
} // повтор вывода символа несколько раз для получения чёткого изо-
бражения

for (repeat=0; repeat<25; repeat++){ // вывод символа «b»
column = 0b00000001;
for (num=0; num<5; num++) {
for(count = num*7;count < (num*7+7);count++){
PORTA = b[count];
PORTB = column;
delay_ms(1);}
column = column<<1;}}

for (repeat=0; repeat<25; repeat++){ // вывод символа «c»
column = 0b00000001;
for (num=0; num<5; num++) {
for(count = num*7;count < (num*7+7);count++){
PORTA = c[count];
PORTB = column;
delay_ms(1);}
column = column<<1;}}
} while(1); } // бесконечный цикл
```

В данном примере управление строками светодиодной матрицы осуществляется с помощью выводов порта PA микроконтроллера, управление столбцами – с помощью выводов порта PB.

В качестве примера рассмотрим последовательный вывод символов «a», «b», «c» на дисплей микросхемы точечного индикатора разрешением 16×16. Двоичный код символов представлен в таблице 2. Точки, которые нужно зажечь на матричном индикаторе, обозначены в таблице нулями, точки, которые не используются – единицами.

Формирование изображения на индикаторе (см. рис. 12) выполним при помощи программы микроконтроллера, написанной на языке программирования C (см. листинг 4).

Запустим в CodeVisionAVR компиляцию кода программы. Затем перейдём в Proteus и укажем в окне свойств микросхемы ATmega16 путь к hex-

файлу на диске компьютера. В поле *CKSEL Fuses* окна свойств установим значение тактовой частоты микроконтроллера 8MHz. При создании проекта в CodeVisionAVR в окне генератора кода на вкладке *Ports Settings* в поле *Direction* для бит Bit 0 – Bit 7 портов Port A и Port B укажем значение *Out* (порты работают на вывод данных).

Двоичный код символов, который записывается в порт PA микроконтроллера, представлен в программе инициализации в виде трёх массивов данных `a[35]`, `b[35]`, `c[35]` размерностью 5×7. Каждой точке символа соответствует отдельный элемент массива, определяющий состояние светодиода матричного индикатора (0 – светодиод включён, 1 – светодиод выключен). Двоичный код, который включает нужные светодиоды активного столбца матрицы, подаётся через порт PA на верхние выводы микросхемы точечного индикатора. Подача значений выполняется программным путём с помощью двух циклов `for` следующим образом. В первом цикле выбирается активный столбец матрицы, после чего во втором цикле выполняются 7 проходов. В каждом проходе в порт PA записывается двоичный код, определяющий состояние одной из 7 точек активного столбца, двоичный код которого записан в порт PB. По окончании 7-го прохода производится выход из второго цикла `for`. Переключение столбцов выполняется при помощи операции `<<1`. Далее 7 проходов второго цикла повторяются для следующего столбца.

Первым в программе определён вывод символа «a». Рисунок отображается на дисплее матрицы начиная с её 7-го столбца. Для получения чёткого изображения на дисплее индикатора повторим вывод символа несколько раз в цикле `for (repeat=0; repeat<25; repeat++)`, где `repeat` – это переменная, которая определяет число повторений. По окончании вывода символа «a» на дисплей матричного индикатора аналогичным образом выполняется вывод символов «b» и «c», после чего все действия повторяются в цикле `do { } while (1);`. В результате на дисплей светодиодной матрицы непрерывно последовательно выводятся символы «a», «b», «c».

В данном примере двоичный код символов подаётся на верхние выводы матрицы побитно, что подразумевает более чёткое изображение по сравнению с побайтным выводом кода. Для вывода изображения шириной 5 точек и высотой 7 точек задействованы 12 контактов точечной матрицы и два порта микроконтроллера. Вывод символов с применением большего числа светодиодов можно организовать, используя большее количество контактов микроконтроллера и матрицы.

Литература

1. ISIS Help, Labcenter Electronics, 2014.

НОВОСТИ МИРА

Индустрия 4.0: RFID «Микрона» контролирует логистику

RFID-решение ГК «Микрон» для автоматизации производственной логистики реализовано на московском высокотехнологичном предприятии по производству медицинского оборудования ООО «С.П. Гелпик». Каждый промаркированный объект теперь можно оперативно отследить на всех этапах логистической цепочки – от начала производства до отгрузки клиенту.

Для контроля перемещения по производственным цехам все объекты (оборудование, детали и маршрутные листы) оснащены радиочастотными метками, входы и выходы оборудованы стационарными считывателями, контролирующими маршрут движения промаркированных объектов. Метки позволяют идентифицировать объект на расстоянии до 12 метров. Данные интегрируются в единую информационную базу для автоматизированного учёта единиц продукции, мониторинга их перемещений, контроля плана работ в режиме реального времени и получения отчётов по соблюдению производственного графика.

«Простая автоматизация логистики позволяет оптимизировать производственный цикл, сократить время простоев и минимизировать возможные ошибки из-за человеческого фактора, обеспечить оперативный контроль за всеми ключевыми процессами и, тем самым, увеличить эффективность производства, – отметил Василий Волосов, руководитель группы развития продуктов RFID и IT ПАО «Микрон». Подобные системы могут быть применены на любых предприятиях, в том числе используется у нас на «Микроне» для маркировки SMIF-контейнеров при производстве микросхем, что позволяет быстро и точно контролировать технический маршрут и последовательность операции».

На сложном технологическом производстве, где требуется ряд последовательных операций в строго определённом порядке с учётом различных переменных факторов (таких как готовность площадок, специалистов, комплектующих и материалов), технические простои, связанные с комплектностью деталей и документов и контролем местонахождения объекта являются одним из основных ресурсов оптимизации производственного цикла. Применение RFID-технологии даёт эффективный инструмент управления производственными процессами и позволяет обеспечить рост операционной эффективности предприятия за счёт точного планирования и контроля операций, сокращения непроизводительных издержек и простоев.

Функционал решения «Микрона» для автоматизации производственной логистики настраивается в зависимости от требований заказчика и решаемых задач. В продуктовой линейке фабрики более 150 видов различных контактных и бесконтактных RFID-меток. Предприятие имеет собственную RFID-лабораторию по разработке и тестированию изделий и полный цикл производства RFID-продукции.

Пресс-служба ПАО «Микрон»

Второй рейтинг крупнейших радиоэлектронных предприятий России

Во второй рейтинг крупнейших радиоэлектронных предприятий России, представленный ЦНИИ «Электроника», попало 50 предприятий, суммарная выручка которых в 2018 году составила 97,9 млрд рублей. Показательно, что объём экспорта предприятий из списка составил 4,3 млрд рублей или менее 5%.

Рейтинг довольно своеобразный: вернее их несколько – для производственных, научно-производственных предприятий и просто организаций, что несколько запутывает ситуацию. К примеру, для производителей электронных компонентов существует отдельный рейтинг из десяти позиций, в котором только две компании являются стопроцентными производителями компонентов («Кулон» и «ММП-Ирбис»), у остальных восьми доля выручки от производства ЭК варьируется от 6% («Кремний») до 87% («Магнетон»).

Новостная рассылка проекта «Мониторинг рынка электроники»