Применение функций обработки строк языка программирования С для управления буквенно-цифровыми дисплеями

Татьяна Колесникова (beluikluk@gmail.com)

В статье рассматривается практическое применение микроконтроллера ATmega16 и строковых функций языка программирования С для управления микросхемами буквенно-цифровых дисплеев из библиотеки компонентов программы Proteus.

Введение

Микроконтроллерная техника является одной из наиболее динамично развивающихся областей современной вычислительной техники. Основой любого микроконтроллерного устройства является заложенная в него программа инициализации, которая задаёт алгоритм функционирования устройства и последовательность выполнения операций. Для написания такой программы используют язык программирования, выбор которого следует выполнять под конкретную задачу.

Система, в которой используется микроконтроллер, может не только чем-то управлять, но и что-то отображать. Одними из основных средств вывода информации для современных цифровых систем являются матричные жидкокристаллические дисплеи (ЖКД), которые бывают следующих видов:

- по типу управляющего интерфейса:
  - с контроллерами – обычно применяются в системах, обладающих не очень большими вычислительными ресурсами; наличие контроллера полностью освобождает разработчика от операции регенерации изображения – ему необходимо только переслать информацию в оперативное запоминающее устройство дисплея – для этого используется 8-разрядный параллельный интерфейс;
  - без контроллеров;
- по типу отображаемой информации:
  - буквенно-цифровые – предназначены для отображения информации в виде букв, цифр, различных знаков; единичные элементы отображения таких индикаторов сгруппированы по строкам и столбцам;
  - графические – отображают информацию, основываясь на матрице точек, равномерно распределённой по всему дисплею.

При производстве буквенно-цифровых жидкокристаллических модулей большинство фирм-производителей применяют контроллер HD44780 [1], который принимает и обрабатывает команды управления и выводит соответствующие символы на дисплей.

Для практического применения дисплея требуется его взаимодействие с внешним источником данных, генератором кодовых комбинаций символов, схемой управления, в качестве которых можно применить микроконтроллер. Каких-либо стандартных правил сопряжения микроконтроллеров с дисплеями не существует, и в каждом конкретном случае сопряжение может выполняться по-разному. Вывод информации на дисплей осуществляют с помощью управляющей программы.

При написании программы для микроконтроллера, который управляет устройством вывода информации, целесообразно использовать язык С, синтаксис которого содержит много функций работы со строками, а также предусматривает возможность их вывода (в данном случае на дисплей).

В языке С программа разбивается на отдельные завершённые логические блоки – процедуры и функции, которые могут объединяться в библиотеки. Применение функций уменьшает код программы, сокращает время её написания, ускоряет внесение изменений. Функцию в любом месте программы можно вызвать на выполнение.

Применение программы Proteus для проектирования устройств вывода информации

Для проектирования устройства вывода информации и моделирования его работы воспользуемся программной средой Proteus [2], библиотека компонентов которой содержит как аналоговые, так и цифровые компоненты, а также микроконтроллеры с возможностью их программирования. В Proteus реализовано большое

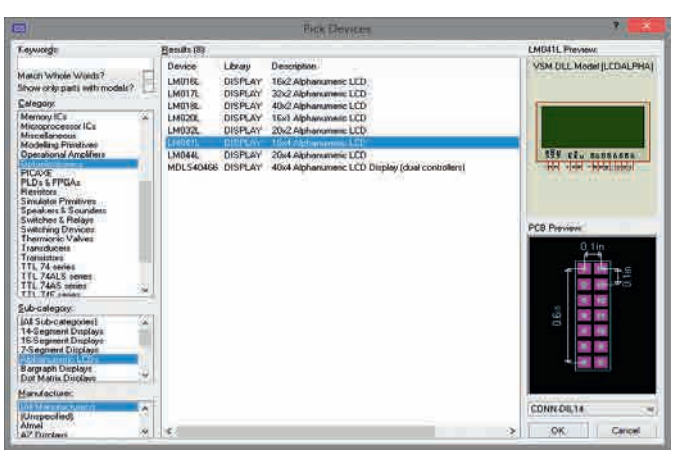

Рис. 1. Раздел Alphanumeric LCDs библиотеки Optoelectronics

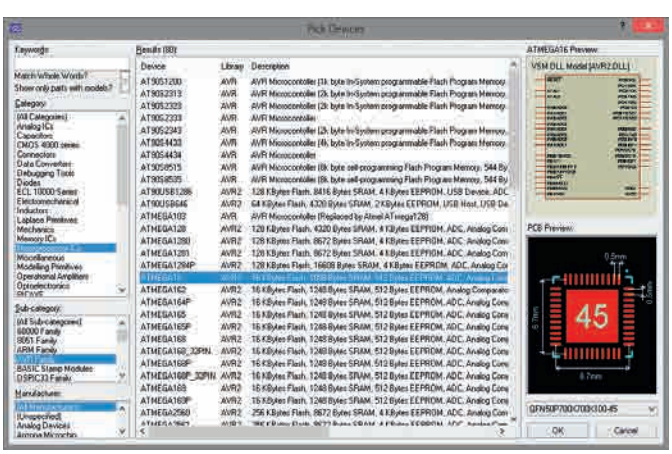

Рис. 2. Раздел AVR Family библиотеки Microprocessor ICs

количество функций для профессионального проектирования микроэлектронных устройств, ориентированных на самые современные средства моделирования. Программа Proteus позволяет автоматизировать все стадии разработки электронных устройств, включая подготовку принципиальных схем, моделирование процессов, происходящих в электронных цепях, компоновку и трассировку печатных плат, редактирование и расширение библиотек компонентов.

В Proteus буквенно-цифровые дисплеи находятся в разделе Alphanumeric LCDs библиотеки Optoelectronics (см. рис. 1) и представлены следующими моделями: LM016L (16×2), LM017L (32×2), LM018L (40×2), LM020L (16×1), LM032L (20×2), LM041L (16×4), LM044L (20×4), MDLS40466 (40×4). В скобках указано разрешение дисплея, например 32×2 – 32 символа × 2 строки. Все микросхемы имеют одинаковую распиновку и отличаются числом строк и числом символов в каждой строке. Для тестирования работы микросхем буквенно-цифровых дисплеев будем использовать микроконтроллер ATmega16. Данная микросхема находится в разделе AVR Family библиотеки Microprocessor ICs (см. рис. 2). Для отладки программного кода выберем компилятор WinAVR.

Буквенно-цифровые дисплеи могут работать в 2 режимах:

- 8-разрядном (для обмена информацией используются выводы D0...D7, при этом информация пересылается за один такт);
- 4-разрядном (для обмена информацией используются выводы D4...D7, при этом информация пересылается за 2 такта – сначала старшие 4 бита, затем младшие 4 бита).

Для подключения микросхемы буквенно-цифрового дисплея к схеме управления используются параллельная синхронная шина данных/команд (D0...D7), вывод выбора операции чтения/записи (RW), вывод выбора регистра данных/команд (RS) и вывод синхронизации (E).

Передача информации между микроконтроллером и микросхемой буквенно-цифрового дисплея в описываемом примере выполняется в 8-разрядном режиме. Отметим, что линии портов микроконтроллера для подключения к указанным выводам дисплея выбираются разработчиком произвольно. Подача управляющих сигналов через подключённые к портам микроконтроллера ATmega16 линии выполняется программно в соответствии с таблицей.

Система команд контроллера HD44780

Команда		Код								
		RS	D7	D6	D5	D4	D3	D2	D1	D0
Очистить дисплей и установить курсор в нулевую позицию		0	0	0	0	0	0	0	0	1
Возврат курсора в нулевую позицию		0	0	0	0	0	0	0	1	-
Выбор направления сдвига курсора при записи следующего символа (I/D=1 – сдвиг вправо, I/D=0 – сдвиг влево); разрешение или запрет сдвига экрана (S=1 – сдвиг разрешён, S=0 – сдвиг запрещён)		0	0	0	0	0	0	1	I/D	S
Включение/выключение дисплея (D=1 – дисплей включён, D=0 – дисплей отключён); включение/выключение отображения курсора на экране (C=1 – курсор отображается, C=0 – курсор не отображается); включение/выключение мигания курсора (B=1 – курсор мигает, B=0 – курсор не мигает)		0	0	0	0	0	1	D	C	B
Сдвиг курсора/экрана (S/C=1 – сдвиг экрана, S/C=0 – сдвиг курсора; R/L=1 – сдвиг вправо, R/L=0 – сдвиг влево)		0	0	0	0	1	S/C	R/L	-	-
Выбор режима работы (DL=1 – 8-разрядный, DL=0 – 4-разрядный); выбор количества используемых для работы строк экрана (N=1 – 2 строки, N=0 – 1 строка); выбор размера отображаемых на экране символов (F=1 – шрифт 5×7 пикс., F=0 – шрифт 5×10 пикс.)		0	0	0	1	DL	N	F	-	-
Выбор адреса (ACG) ячейки памяти CGRAM		0	0	1	ACG	ACG	ACG	ACG	ACG	ACG
Выбор адреса (ADD) ячейки памяти DDRAM		0	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD
Запись данных в выбранную ячейку памяти DDRAM или CGRAM	8-разрядный режим	1	8 бит	7 бит	6 бит	5 бит	4 бит	3 бит	2 бит	1 бит
	4-разрядный режим	1	8 бит	7 бит	6 бит	5 бит	-	-	-	-
			4 бит	3 бит	2 бит	1 бит	-	-	-	-

Необходимо отметить, что контроллер HD44780 содержит 3 вида памяти:

1. DDRAM – оперативное запоминающее устройство, в котором хранятся коды символов, отображаемых на экране.
2. CGROM – постоянное запоминающее устройство, которое содержит «битовое изображение» символов.
3. CGRAM – оперативное запоминающее устройство, является частью CGROM, предназначено для хранения символов пользователя.

Последовательность действий, которые необходимо выполнить управляющей схеме при совершении операции записи по 8-разрядной шине, может быть следующей:

1. Установить значение линии RW=0 (запись в микросхему буквенно-цифрового дисплея).
2. Установить значение линии RS=0 (приём команд).
3. Вывести значение байта команды 00001111 на линии шины D0...D7 (команда включения дисплея).
4. Вывести значение байта команды 00110100 на линии шины D0...D7 (установка разрядности шины).
5. Вывести значение байта команды 00000001 на линии шины D0...D7 (очистка дисплея и установка курсора в нулевую позицию).
6. Установить значение линии RS=1 (приём данных).
7. Вывести значение байта данных на линии шины D0...D7.

Необходимо учитывать, что большинство операций, выполняемых контроллером, занимают значительное время, около 40 мкс, а время выполнения некоторых доходит до единиц миллисекунд, поэтому в программе управления жидкокристаллическим модулем совершению любой операции должны предшествовать команды задержки. Также необходимо обеспечить формирование тактового сигнала на линии E микросхемы буквенно-цифрового дисплея. Сделать это можно программно посредством чередования подачи значений нуля и единицы.

Применение функций работы со строками языка C для управления буквенно-цифровыми дисплеями

Язык C поддерживает множество функций обработки строк. Наиболее типичные операции над строковыми данными оформлены в виде функций стандартной библиотеки работы со строками, подключаемой к программе с помощью заголовочного файла

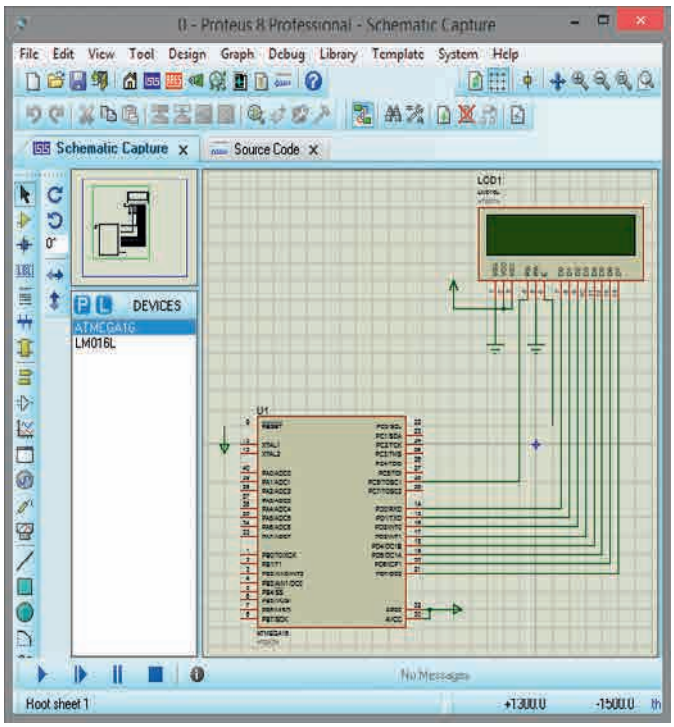

Рис. 3. Подключение микросхемы LM016L к схеме управления

Листинг 1

```
#include <inttypes.h> // подключение заголовочных
файлов
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <util/delay.h>
#include <string.h> // заголовочный файл для работы
с функциями обработки строк

void e ( ) // подпрограмма формирования тактового
сигнала на линии E дисплея
{PORTC |= 1<<0; // установка «1» на выводе PC0
микроконтроллера
_delay_ms(200); // задержка
PORTC &=~ (1<<0); // установка «0» на выводе PC0
микроконтроллера
_delay_ms(200); } // задержка

int main() // начало программы
{ int len, i;
char s[]="New String"; // текст строки для вывода
на дисплей
DDRD=DDRC=0xff; // инициализация портов PD и PC
PORTD=PORTC=0x00; // порты PD и PC работают на вы-
вод данных
e ( ); PORTC &=~ (1<<6); // RS=0 (приём команд)
e ( ); PORTD=0b00001111; // включение дисплея
e ( ); PORTD=0b00110100; // установка 8-разрядной
шины
e ( ); PORTD=0b00000001; // очистка дисплея и уста-
новка курсора в нулевую позицию

len=strlen(s); // определение длины строки
// и запись полученного значения в переменную len
e ( ); PORTC |= 1<<6; // RS=1 (приём данных)
for (i=0;i<len;i++) // цикл вывода двоичного кода
символов строки в порт PD
{PORTD=s[i]; // запись в порт PD двоичного кода
очередного символа строки
e ( ); // вызов подпрограммы формирования тактового
сигнала на линии E дисплея
}}
```

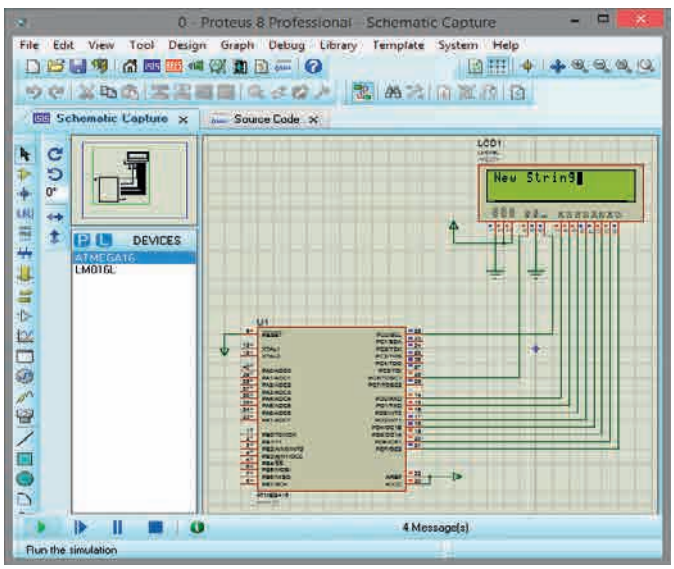

Рис. 4. Результат работы программы управления буквенно-цифровым дисплеем

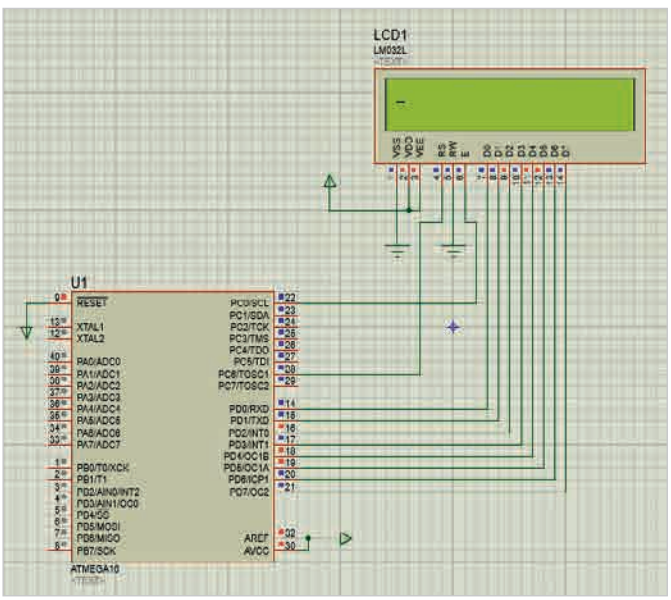

Рис. 5. Подключение микросхемы LM032L к схеме управления

`string.h`. Синтаксис подключения файла следующий: `#include <string.h>`.

Символьная строка представляет собой набор из одного или более символов. В языке С нет специального типа данных, который можно было бы использовать для описания строк, поэтому строки представляют в виде массива элементов типа `char`. Символы строки размещаются в памяти в соседних ячейках по одному. При этом последний элемент массива – это символ `'\0'`, который в языке С используется для определения конца строки. Необходимо отметить, что нулевой символ – это не цифра 0, он не выводится на печать. Наличие нулевого символа предусматривает, что количество ячеек массива должно быть на одну больше, чем число символов, которые необходимо разместить в памяти. К примеру, определение переменной `char str[10]` предусматривает, что строка может содержать максимум 9 символов.

Стандартная библиотека языка программирования С содержит класс функций для работы со строками. Для определения длины строки используется функция `strlen()`, которая возвращает число символов, при этом завершающий нулевой символ не учитывается. Функции `strcpy()` и `strncpy()` применяют для копирования строк, а функцию `strdup()` – для дублирования строки.

Применение функции *strlen()*

Рассмотрим работу с функцией `strlen()` на примере дисплея LM016L, разрешение которого составляет 16×2.

LM016L имеет 14 контактов, назначение которых следующее:

- VSS – GND;
- VDD – напряжение питания +5 В;
- VEE – напряжение контрастности от 0 до +5 В (настройка контрастности отображаемых на дисплее символов);
- RS – выбор регистра данных DR (RS=1) или команд IR (RS=0);

- RW – выбор операции чтения (RW=1) или записи (RW=0);
- E – линия синхронизации;
- D0...D7 – шина данных/команд.
- Необходимо отметить, что все дисплеи на основе контроллера HD44780 имеют схожую распиновку.

Подсоединим выводы модуля дисплея D0...D7 к выводам порта PD0...PD7, а выводы RS и E к выводам PC6 и PC0 порта микроконтроллера ATmega16, как показано на рисунке 3. Вывод RW подключим к «земле», поскольку в данной системе будет выполняться только запись информации в микросхему LM016L. Выводы VSS и VDD подключим к «земле» и напряжению +5 В соответственно. На вывод VEE подаётся напряжение контрастности (от 0 до +5 В). На практике этот вывод подключается к питанию через подстроечный резистор, который позволяет плавно регулировать контрастность отображения символов на дисплее.

Используя систему команд контроллера HD44780, напишем на языке С программу для микроконтроллера ATmega16, которая в качестве примера будет выводить на экран дисплея заданную строку (см. листинг 1).

Введём код программы на вкладке Source Code схемного проекта и запустим моделирование. На экран микросхемы LM016L будет посимвольно выведена строка определённой длины, двоичные коды символов которой были поданы на шину D0...D7. Результат представлен на рисунке 4.

Применение функций *strcpy()* и *strncpy()*

Используя функции `strcpy(s1, s2)` и `strncpy(s1, s3, 5)` с указанными параметрами, скопируем строку s2 в строку s1 и выведем полученный текст с 10-й позиции 1-й строки, а также скопируем 5 символов из строки `s3` в строку `s1` и выведем полученный текст с 8-й позиции 2-й строки дисплея LM032L. Подключение дисплея к схеме управления представлено на рисунке 5. Текст программы на языке С приведён в листинге 2.

Изначально в программе строковым переменным присвоены следующие значения: `char s1[]="Stroka "`, `char s2[]="Vuvod texta"`, `char s3[]="Obzor knigi"`. После выполнения функции `strcpy(s1, s2)` в переменную `s1` заносится текст `Vuvod texta`. При этом текст `Stroka`, который хранился в переменной `s1`, затирается. Таким образом, в цикле `for (i=0;i<len1;i++) { PORTD=s1[i]; e ( );}` на экран первой строки дисплея выводится текст `Vuvod texta` (см. рис. 6а). После выполнения функции `strncpy(s1, s3, 5)` в переменную `s1`, в которой хранится текст `Vuvod texta`, копируется 5 символов из строки `s3` (символы `Obzor`). При этом 5 символов текста `Vuvod`, которые хранились в переменной `s1`, затираются. Таким образом, в цикле `for (i=0;i<len2;i++) {PORTD=s1[i]; e ( );}` на экран во 2-ю строку дисплея выводится текст `Obzor texta` (см. рис. 6б).

Листинг 2

```
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <util/delay.h>
#include <string.h>

void e ( )
{PORTC |= 1<<0; // установка «1» на выводе PC0 микроконтроллера
_delay_ms(200); // задержка
PORTC &=~ (1<<0); // установка «0» на выводе PC0 микроконтроллера
_delay_ms(200); } // задержка

int main()
{int len1, len2, i;
char s1[]="Stroka"; // строка 1
char s2[]="Vuvod texta"; // строка 2
char s3[]="Obzor knigi"; // строка 3
DDRD=DDRC=0xff; // инициализация портов PD и PC
PORTD=PORTC=0x00; // порты PD и PC работают на вывод данных
e ( ); PORTC &=~ (1<<6); // RS=0 (приём команд)
e ( ); PORTD=0b00001111; // включение дисплея
e ( ); PORTD=0b00110100; // установка 8-разрядной шины
e ( ); PORTD=0b00000001; //очистка дисплея и установка курсора в нулевую позицию
e ( ); PORTD=0b00111000; // установка 2-строчного режима
e ( ); PORTD=0b10001001; // выбор адреса ячейки памяти DDRAM (10 позиция 1 строки)
e ( ); PORTC |= 1<<6; // RS=1 (приём данных)

strcpy(s1,s2); // копирование строки s2 в строку s1
len1=strlen(s1); // определение длины строки s1
for (i=0;i<len1;i++)
{ PORTD=s1[i]; e ( );} // вывод строки Vuvod texta на экран дисплея

PORTC &=~ (1<<6); // RS=0 (приём команд)
e ( ); PORTD=0b11000111; // выбор адреса ячейки памяти DDRAM (8 позиция 2 строки)
e ( ); PORTC |= 1<<6; // RS=1 (приём данных)

strncpy(s1,s3,5); // копирование 5 символов строки s3 в строку s1
len2=strlen(s1); // определение длины строки s1
for (i=0;i<len2;i++)
{ PORTD=s1[i]; // вывод строки Obzor texta на экран дисплея
e ( );}} // вызов подпрограммы формирования тактового сигнала на линии E дисплея
```

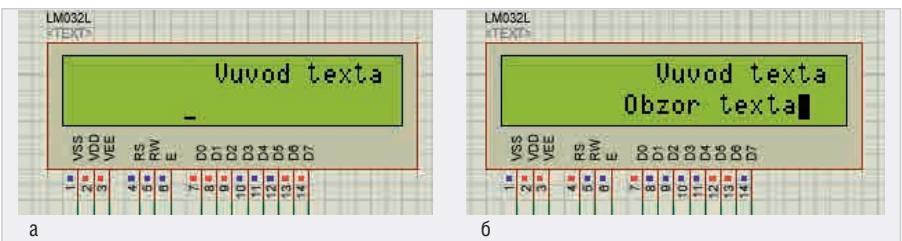

Рис. 6. Результат применения функций: а) `strcpy(s1, s2)`; б) `strncpy(s1, s3, 5)`

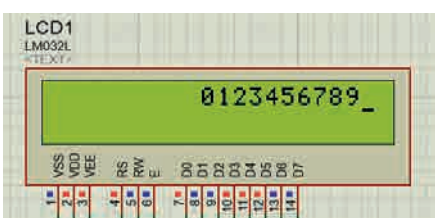

Рис. 7. Результат применения функции *strdup()*

Применение функции *strdup()*

Функция `strdup()` выполняет дублирование строк с выделением памяти под новую строку. Синтаксис функции: `char *strdup(const char *str)`, где `str` – указатель на дублируемую строку. В представленном примере функция `strdup()` дублирует строку, на которую указывает аргумент `str`, результат выводится на экран дисплея LM032L, начиная с 10-й позиции 1-й строки (см. рис. 7). Память под дубликат строки выделяется с помощью функции `malloc()` и по окончании работы с дубликатом очищается с помощью функции `free()`. Пример программы приведён в листинге 3.

Применение функции *strtok()*

При выводе информации на экран дисплея может быть полезной функция `strtok()`, назначение которой – разбиение заданной строки на отдельные части по указанному разделителю.

Синтаксис функции: `char *strtok(const char *str, const char *sep)`, где `str` – указатель на разбиваемую строку, `sep` – указатель на строку, содержащую символ-разделитель. При первом вызове функции в качестве первого параметра указывается строка, которую требуется разбить, вторым параметром указывается стро-

Листинг 3

```
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <util/delay.h>
#include <string.h> // заголовочный файл для работы со строковыми функциями
#include <stdlib.h> // подключение заголовочного файла для работы с функцией free

void e ( )
{PORTC |= 1<<0; // установка «1» на выводе PC0 микроконтроллера
_delay_ms(200); // задержка
PORTC &=~ (1<<0); // установка «0» на выводе PC0 микроконтроллера
_delay_ms(200);} // задержка

int main()
{ int len, i;
DDRD=DDRC=0xff; // инициализация портов PD и PC
PORTD=PORTC=0x00; // порты PD и PC работают на вывод данных
e ( ); PORTC &=~ (1<<6); // RS=0 (приём команд)
e ( ); PORTD=0b00001111; // включение дисплея
e ( ); PORTD=0b00110100; // установка 8-разрядной шины
e ( ); PORTD=0b00000001; //очистка дисплея и установка курсора в нулевую позицию
e ( ); PORTD=0b00111000; // установка 2-строчного режима
e ( ); PORTD=0b10001001; // выбор адреса ячейки памяти DDRAM (10 позиция 1 строки)
e ( ); PORTC |= 1<<6; // RS=1 (приём данных)

char str[]="0123456789"; // исходная строка
char *str2; // переменная, в которую будет помещён указатель на дубликат строки
str2=strdup(str); // дублирование строки str
len=strlen(str2); // определение длины строки str2
for (i=0;i<len;i++)
{ PORTD=str2[i]; // вывод строки str2 на экран дисплея
e ( ); } // вызов подпрограммы формирования тактового сигнала на линии E дисплея
free (str2); } // очистка памяти, выделенной под дубликат строки
```

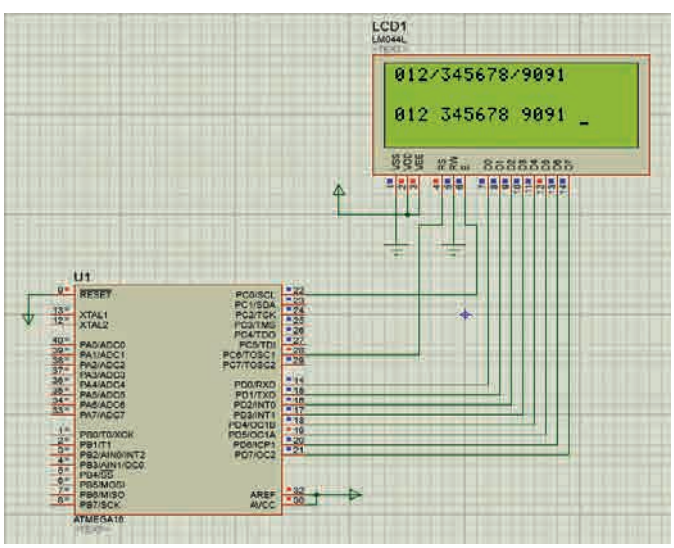

Рис. 8. Результат применения функции *strtok()*

Листинг 4

```
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <util/delay.h>
#include <string.h>

void e ( )
{PORTC |= 1<<0; // установка «1» на выводе PC0 микроконтроллера
_delay_ms(200); // задержка
PORTC &=~ (1<<0); // установка «0» на выводе PC0 микроконтроллера
_delay_ms(200);} // задержка

int main()
{ int len, i;
DDRD=DDRC=0xff; // инициализация портов PD и PC
PORTD=PORTC=0x00; // порты PD и PC работают на вывод данных
e ( ); PORTC &=~ (1<<6); // RS=0 (приём команд)
e ( ); PORTD=0b00001111; // включение дисплея
e ( ); PORTD=0b00110100; // установка 8-разрядной шины
e ( ); PORTD=0b00000001; //очистка дисплея и установка курсора в нулевую позицию
e ( ); PORTD=0b00111000; // установка 2-строчного режима
e ( ); PORTD=0b10000000; // выбор адреса ячейки памяти DDRAM (1 позиция 1 строки)
e ( ); PORTC |= 1<<6; // RS=1 (приём данных)

char str[]="012/345678/9091"; // исходная строка
char razdelitel[]="/"; // разделитель
char probel[]=" "; // пробел
char *str2; // переменная, в которую будет заноситься указатель на новый фрагмент строки

len=strlen(str); // определение длины строки str
for (i=0;i<len;i++)
{ PORTD=str[i]; e ( );} // вывод исходной строки символов на первую строку дисплея

PORTC &=~ (1<<6); // RS=0 (приём команд)
e ( ); PORTD=0b10010100; // выбор адреса ячейки памяти DDRAM (1 позиция 3 строки)
e ( ); PORTC |= 1<<6; // RS=1 (приём данных)

str2=strtok(str,razdelitel); // выделение первого фрагмента строки str
while (str2 != NULL) // выделение следующих фрагментов
{len=strlen(str2); // определение длины очередного фрагмента

for (i=0;i<len;i++)
{ PORTD=str2[i]; e ( ); } // посимвольный вывод фрагмента на 3 строку дисплея
PORTD=probel[0]; e ( ); // отделение фрагментов строки пробелом
str2=strtok (NULL,razdelitel); // выделение очередного фрагмента строки
}}
```

ка-разделитель. При последующих вызовах функции для этой же строки первым параметром должен быть `NULL`.

В представленном примере (см. листинг 4) строка `012/345678/9091` разбивается на части по разделителю «/». Исходный набор символов выводится на экран дисплея LM044L, начиная с 1-й позиции 1-й строки, затем функция `strtok()` определяет первый фрагмент в исходной строке, далее при помощи функции `strlen()` выполняются определение длины этого фрагмента и его посимвольный вывод начиная с 3-й строки дисплея в цикле `for (i=0;i<len;i++)`. Функция `strtok()` выделяет фрагменты по разделителю «/», пока не будет достигнут конец строки. В данном примере фрагменты новой строки разделены пробелами.

Отметим, что система команд контроллера HD44780 предусматривает установку 1- и 2-строчного режимов и для вывода символов на все 4 строки дисплея LM044L используется 2-строчный режим.

Например, чтобы задать вывод на дисплей LM044L двух строк (каждой с новой строки) начиная, к примеру, с 1-го бита 1-й и 3-й строк дисплея, необходимо в режиме приёма команд (RS=0) подать на 8-разрядную шину микросхемы LM044L следующие команды:

- D7=0, D6=0, D5=1, D4=1, D3=1, D2=0, D1=0, D0=0 (установка 2-строчного режима);
- D7=1, D6=x6, D5=x5, D4=x4, D3=x3, D2=x2, D1=x1, D0=x0 (выбор адреса ячейки памяти DDRAM, где значения x6...x0 – это двоичный код адреса ячейки памяти DDRAM).

Результат моделирования схемы представлен на рисунке 8.

Литература

1. HD44780U (LCD-II) (Dot Matrix Liquid Crystal Display Controller/Driver). Hitachi, Ltd. 1998.
2. ISIS Help, Labcenter Electronics, 2014.

НОВОСТИ МИРА

USB-платформа серии Streamline: расширенные возможности без компромиссов

Keysight Technologies представляет новую измерительную платформу, гарантирующую стабильность и воспроизводимость результатов измерений на всех этапах разработки. Новая серия Streamline состоит из компактных USB-приборов: векторных анализаторов цепей (VNA), осциллографов и генераторов сигналов произвольной формы (AWG), которые используют проверенные временем технологии, алгоритмы измерения и прикладное программное обеспечение Keysight.

Измерительная платформа Streamline позволяет разработчикам в условиях жёсткой конкурентной борьбы обеспечить эффективность, оптимизацию ресурсов и соблюдение сжатых сроков при создании новых электронных устройств. Теперь заказчики могут быть уверены, что при переходе от одного этапа разработки к другому они получат точные и воспроизводимые результаты измерений независимо от типа используемого прибора – USB, модульного или настольного.

Управление новыми приборами Keysight осуществляется с ПК через интерфейс USB, за счёт чего экономится место на рабочем столе. Кроме того, такими приборами могут пользоваться все члены группы разработчиков. USB-приборы занимают мало места в стойке, поэтому они идеальны для ручного или полуавтоматического тестирования при выполнении квалификационных испытаний и в мелкосерийном производстве.

Платформа Keysight серии Streamline объединяет следующие модели:

- Компактные двухпортовые векторные анализаторы цепей серии P937xA с диапазоном частот до 26,5 ГГц предназначены для тестирования пассивных устройств, таких как антенны, фильтры и дуплексеры. При работе на внешнем ПК контекстно ориентированный интерфейс пользователя идентичен интерфейсу последних моделей настольных векторных анализаторов цепей Keysight.
- Высокопроизводительные осциллографы серии P924xA обеспечивают полный набор измерительных функций, а также расширенные возможности запуска, высокую скорость обновления сигналов на экране и такие востребованные функции, как запуск по выделенной зоне. При работе на пользовательском ПК используется интерфейс осциллографа Keysight InfiniiVision, аналогичный интерфейсу настольных осциллографов.

- Трёхканальный генератор сигналов произвольной формы P9336A, использующий уникальную технологию Trueform, обеспечивает разрешение 16 разрядов с максимальной полосой анализа 540 МГц и максимальной встроенной памятью 4 ГБ. Области применения – от стандартного тестирования до генерации сложных сигналов I/Q для получения характеристик приёмопередатчиков и модуляторов. Создание сигналов упрощено за счёт совместимости USB-генератора сигналов произвольной формы с ПО Signal Studio.

Пресс-служба Keysight Technologies