УДК 004.41

О. В. КЛЕМЕШЕВА
КГТУ ИМ. И. РАЗЗАКОВА, ТФ ИМ. Х. А. РАХМАТУЛИНА,
ТОКМОК, КЫРГЫЗСКАЯ РЕСПУБЛИКА
E-MAIL: OLGAKL_86@MAIL.RU
O. V. KLEMESHOVA
KSTU N. A. I.RAZZAKOV TB N.A. H.A.RAHMATULIN,
TOKMOK, KYRGYZ REPUBLIC

Ж. Б. МЕКЕНБАЕВА
КГУСТА ИМ. Н. ИСАНОВА,
БИШКЕК, КЫРГЫЗСКАЯ РЕСПУБЛИКА
J. B. MEKENBAEVA
KSUCTA N.A. N. ISANOV,
BISHKEK, KYRGYZ REPUBLIC

Т. АСКАРБЕК У.
КГУСТА ИМ. Н. ИСАНОВА,
БИШКЕК, КЫРГЫЗСКАЯ РЕСПУБЛИКА
T. ASKARBEK U.
KSUCTA N.A. N. ISANOV,
BISHKEK, KYRGYZ REPUBLIC

E.mail. ksucta@elcat.kg

МИКРОКОНТРОЛЛЕРЫ В СИСТЕМАХ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ

MICROCONTROLLERS IN THE SYSTEMS OF AUTOMATIC CONTROL

Макалада микроконтроллерлорду программалоодогу актуалдуу маселелер каралды. Микроконтроллерлорду программалоонун жардамы менен аппараттык техникалардын көп практикалык маселелерин чыгарса болот. Алардын негиздери жана практикалык колдонуулары талданат.

Чечүүчү сөздөр: *микроконтроллер, микросхема, регистр, компилятор, диод.*

В статье рассмотрены актуальные вопросы программирования микроконтроллеров. С помощью программирования микроконтроллеров можно решить многие практические задачи аппаратной техники. Анализируется их сущность и практическое применение.

Ключевые слова*: микроконтроллер, микросхема, регистр, компилятор, диод.*

In article topical issues of programming of microcontrollers are considered. By means of programming of microcontrollers it is possible to solve many practical problems of the hardware equipment. Their essence and practical application is analyzed.

Key words*: microcontroller, chip, register, compiler, diode.*

Введение. Микроконтроллер – это микросхема (рис. 1.), предназначенная для управления электронными устройствами. Типичный микроконтроллер сочетает на одном кристалле функции процессора и периферийных устройств, содержит ОЗУ и (или) ПЗУ. По сути, это однокристальный компьютер, способный выполнять относительно простые задачи.

Цель исследования. На основе практического примера показать преимущественные характеристики использования микроконтроллеров, необходимости их внедрения в различные устройства.

Рис. 1. Микроконтроллер

Подключение и управление микроконтроллером. Микроконтроллеры AVR оснащены Гарвардской архитектурой. Каждая из областей памяти располагаются в своем адресном пространстве. Память данных в контроллерах осуществляется посредством регистровой, энергонезависимой и оперативной памяти. Регистровая память предусматривает наличие 32 регистров общего назначения, которые объединены в файл, а также служебные регистры для ввода и вывода. И первые, и вторые располагаются в пространстве ОЗУ, однако не являются его частью.

Микроконтроллеры AVR являются простыми в использовании, имеют низкую потребляемую мощность и высокий уровень интеграции. Как правило, такие микроконтроллеры могут использоваться на самых разных устройствах, в том числе системах общего назначения, системах оповещения, для ЖК-дисплеев, плат с ограниченным пространством.

Промышленность выпускает следующие виды микроконтроллеров:

- встраиваемые;
- 8-, 16- и 32-разрядные;
- цифровые сигнальные процессоры.

Микроконтроллер объединяет противоречивые эксплуатационные показатели: габариты, мощность и цена изделия. Поэтому до сих пор используются 8-разрядные модели. Они обладают довольно низкой производительностью, но позволяют экономить энергоресурсы. Цифровые сигнальные процессоры способны обрабатывать в реальном времени большие потоки данных, однако их стоимость намного выше.

Количество используемых кодов операций может быть неодинаковым. Поэтому применяются системы команд RISC и CISC. Первая считается сокращенной и выполняется за один такт генератора. Это позволяет упростить аппаратную реализацию ЦП, повысить производительность микросхемы. CISC — сложная система, способная значительно увеличить эффективность устройства. Одной из наиболее популярных в электронной промышленности является продукция компании Atmel, построенная на базе RISC-ядра.

Программирование микроконтроллеров. По своей структуре языки программирования микроконтроллеров мало отличаются от тех, что используются для персональных компьютеров. Современные микроконтроллеры в основном используют C++ и Ассемблер.

Низкоуровневый Ассемблер использует прямые инструкции, обращенные непосредственно к чипу. Поэтому от программиста требуется безукоризненное знание системных команд процессора. Написание программного обеспечения на Ассемблере занимает значительное время. Главным преимуществом языка является высокая скорость исполнения готовой программы.

Для программирования микроконтроллеров можно использовать практически любые языки программирования. Но популярнее всех C++. Это язык высокого уровня,

позволяющий работать с максимальным комфортом. Поэтому микросхемы производ… «Atmel» адаптированы именно к этому языку.

C++ — это гармоничное сочетание низкоуровневых и высокоуровневых возможностей. Поэтому в код можно внедрить вставки на Ассемблере. Готовый программный продукт легко читается и модифицируется. Скорость разработки достаточно высокая. При этом доскональное изучение архитектуры МК и системы команд ЦП не требуется. Компиляторы C++ снабжаются библиотеками внушительного размера, что облегчает работу программиста.

Нужно отметить, что выбор оптимального языка программирования зависит также от аппаратного обеспечения. При малом количестве оперативной памяти использовать высокоуровневый C++ нецелесообразно. В данном случае больше подойдет Ассемблер. Он обеспечивает максимальное быстродействие за счет короткого кода программы.

Исследование. Для программирования микроконтроллеров систем автоматичекого управления был выбран язык программирования C++. У языка C++ есть сильная сторона это переносимость кода.

Выбор компилятора. Для AVR существует множество разных компиляторов Си: В первую очередь это **IAR AVR C** — почти однозначно признается лучшим компилятором для AVR, т.к. сам контроллер создавался в тесном сотрудничестве Atmel.

Вторым идет **WinAVR GCC** — мощный оптимизирующий компилятор. Еще он отлично интегрируется в AVR Studio позволяя вести отладку прямо там, что очень удобно. Также есть **CodeVision AVR C**, ставший очень популярным компилятором в связи со своей простотой. Рабочую программу в нем получить можно уже через несколько минут — этому сильно способствует мастер стартового кода.

Исследуя все доступные компиляторы мы выбрали компилятор WinAVR и AVR Studio.

Микроконтроллер будет работать следующим образом:

При приходе по COM-порту единицы (код 0x31) будем открывать диод, а при приходе нуля (код 0x30) закрывать. Причем, все будет сделано на прерываниях, а фоновой задачей будет переключение другого диода.

Сборка схемы. Нам надо соединить модуль USB-USART конвертера с выводами USART микроконтроллера. То есть Rx контроллера соединяем с Tx конвертера, а Tx конвертера с Rx контроллера. Полученная схема представлена на рис. 2.

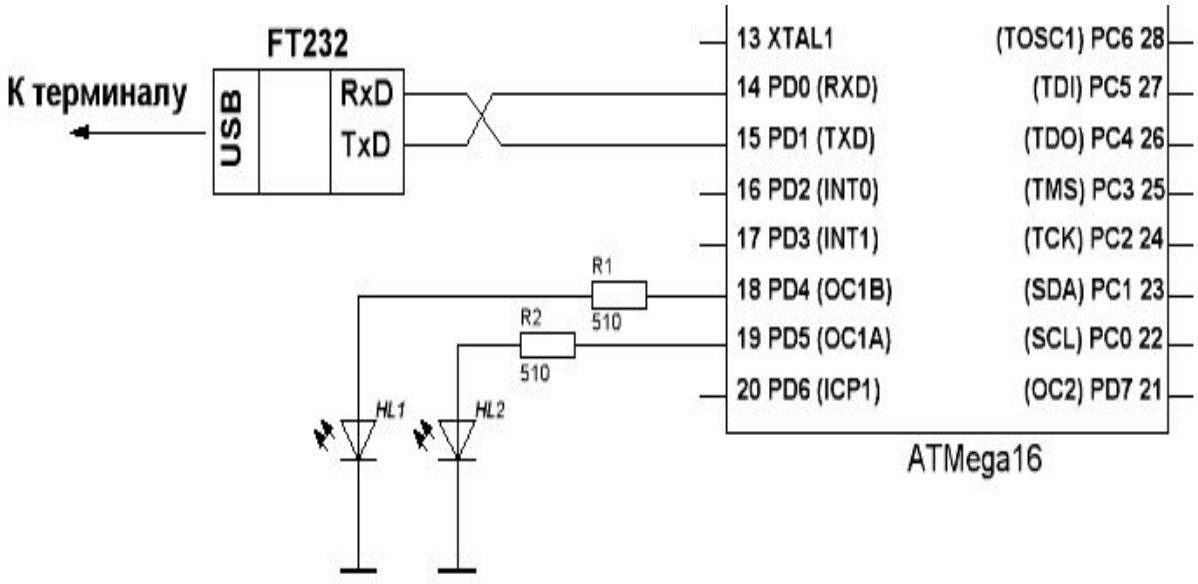

Рис. 2. Схема микроконтроллера

Программирование кода. Вначале добавляются нужные библиотеки и заголовки с

определениями. Т.к. C++ –универсальный язык, то ему надо указать, что мы работаем именно с AVR, поэтому вызывается заголовочный файл avr/io.h. Этот файл находится в папке **WinAVR**, и в нем содержится описание всех регистров и портов контроллера.

Программа на языке C++ состоит, главным образом, из функций. Они могут быть вложенными и вызываться друг из друга в любом порядке и разными способами. Каждая функция имеет три обязательных параметра:

- Возвращаемое значение.
- Передаваемые параметры.
- Тело функции.

Вначале инициализируются порты и UART.

```
int main(void)
{
unsigned char i;
#define XTAL 8000000L
#define baudrate 9600L
#define bauddivider (XTAL/(16*baudrate)-1)
#define HI(x) ((x)>>8)
#define LO(x) ((x)& 0xFF)
UBRRL = LO(bauddivider);
UBRRH = HI(bauddivider);
UCSRA = 0;
UCSRB = 1<<RXEN|1<<TXEN|1<<RXCIE|0<<TXCIE;
UCSRC = 1<<URSEL|1<<UCSZ0|1<<UCSZ1;
}
```

Для определения макросов препроцессора используется директива **#define**. Макросы облегчают рутинные операции по вычислению нужных коэффициентов. В первой строке мы говорим, что вместо **XTAL** можно подставлять 8000000, а **L** — указание типа, long — это тактовая частота процессора. То же самое **baudrate** — частота передачи данных по UART.

bauddivider - вместо него будет подставляться выражение, вычисленное по формуле из двух предыдущих. **LO** и **HI** из этого результата возьмут младший и старший байты, т.к. в одного байта может оказаться недостаточно. В **HI** делается сдвиг икса (входной параметр макроса) восемь раз вправо, в результате от него останется только старший байт. А в **LO** мы делаем побитовое И с числом 00FF, в результате останется только младший байт.

Запись вида **1<<RXEN** означает следующее: взять 1 и поставить ее на место **RXEN** в байте. **RXEN** это 4-й бит регистра **UCSRB**, поэтому **1<<RXEN** образует двоичное число 00010000, **TXEN** — это 3-й бит, а **1<<TXEN** дает 00001000. Одиночная «|» это побитовое **ИЛИ**, поэтому 00010000 | 00001000 = 00011000. В итоге, собранное число записывается в UCSRB.

Отладка. После запуска появляется указатель, показывающий, какую команду выполняет процессор в текущий момент (рис. 3).

```
#include <avr/io.h>


int main(void)
{
unsigned char i;

#define XTAL 8000000L
#define baudrate 9600L
#define bauddivider (XTAL/(16*baudrate)-1)
#define HI(x) ((x)>>8)
#define LO(x) ((x)& 0xFF)

UBRRL = LO(bauddivider);
UBRRH = HI(bauddivider);
UCSRA = 0;
UCSRB = 1<<RXEN|1<<TXEN|1<<RXCIE|0<<TXCIE;
UCSRC = 1<<URSEL|1<<UCSZ0|1<<UCSZ1;

return 0;
}
```

Рис. 3. Отладка программы

Инструкции выполнены. Если зайти в дерево **I/O View**, в раздел USART и проанализировать там UBRRL, то видим, что он принял значение 0x33 (рис. 4).

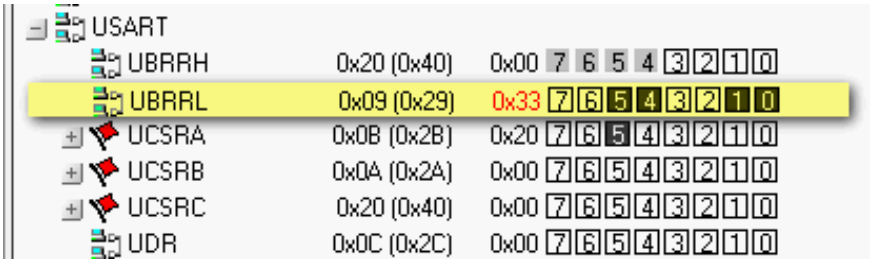

Рис. 4. Проверка содержимого регистра

Анализ содержимого другого регистра показал, что все указанные биты выставляются, причем, выставляются одновременно для всего байта.

Для зажигания LED1 необходимо выставить бит 4 в DDRD в единицу и дальше работать битом 4 в регистре PORTD, для LED2 – то же самое, только с битом 5. Поэтому добавляем после инициализации UART инициализацию нашего порта.

```
#define LED1 4
#define LED2 5
#define LED_PORT PORTD
#define LED_DDR DDRD
LED_DDR = 1<<LED1;
```

В цикл добавим команды установки и сброса бита в порту. Именно им будет определяться уровень напряжения на выходе:

```
while(1)
 {
 i++;
 LED_PORT=0<<LED1;
 LED_PORT=1<<LED1;
 }
```

В результате получается следующий код:

```
#include <avr/io.h>
```

```
int main(void)
{
volatile unsigned char i;
#define XTAL 8000000L
#define baudrate 9600L
#define bauddivider (XTAL/(16*baudrate)-1)
#define HI(x) ((x)>>8)
#define LO(x) ((x)& 0xFF)
UBRRL = LO(bauddivider);
UBRRH = HI(bauddivider);
UCSRA = 0;
UCSRB = 1<<RXEN|1<<TXEN|1<<RXCIE|0<<TXCIE;
UCSRC = 1<<URSEL|1<<UCSZ0|1<<UCSZ1;
#define LED1 4
#define LED2 5
#define LED_PORT PORTD
#define LED_DDR DDRD
LED_DDR = 1<<LED1;
while(1)
{
i++;
LED_PORT=0<<LED1;
LED_PORT=1<<LED1;
}
return 0;
}
```

Теперь можно производить моделирование реального микроконтроллера.

Заключение. В ходе исследования выявлено следующее:

Микроконтроллер представляет собой специальную микросхему, которая предназначена для выполнения некоего набора сложных функций по управлению электронным устройством.

Микропроцессорные технологии очень эффективны. Одно и то же устройство, которое раньше собиралось на традиционных элементах, будучи собрано с применением микропроцессора становится проще, не требует регулировки и меньше по размерам. С применением микроконтроллеров появляются практически безграничные возможности по добавлению новых потребительских функций и возможностей.

Список литературы

1. Белов А.В. Программирование микроконтроллеров для начинающих и не только…[Текст] / А.В.Белов. — СПб.: Наука и техника, 2016. — 352 с.

2. Кравченко А.В. 10 практических устройств на AVR-микроконтроллерах. Книга 2 [Текст] / А.В.Кравченко. – К.: “МК-Пресс”, СПб.: “КОРОНА-ВЕК”, 2009. – 320 с.

3. Шпак Ю.А. Программирование на языке С для AVR- и PIC-микроконтроллеров [Текст] / Ю.А.Шпак. - К.: “МК-Пресс”, 2006. – 400 с.