Современная среда разработки mikroC для программирования микроконтроллеров на языке высокого уровня Си

(часть 2)

Олег Вальпа (Челябинская обл.)

Приводится описание современной, мощной и удобной среды разработки mikroC, которая включает большую библиотеку готовых функций для работы с разнообразными интерфейсами и устройствами и позволяет быстро создавать эффективные программы на языке высокого уровня Си для микроконтроллеров семейств PIC, AVR, MCS-51 и др.

Библиотека функций с примерами программ и схем подключения

Рассмотрим описание основных библиотечных функций для работы с наиболее распространёнными узлами микроконтроллера и примеры их использования в программе на основе приводимых схем подключения.

Функции для работы с ADC

Модуль ADC (Analog to Digital Converter, АЦП) есть во многих моделях PIC-контроллеров. Библиотечная функция Adc_Read (см. таблицу 3) предназначена для удобства работы с этим модулем.

Следующая программа читает аналоговое значение из второго канала АЦП и выводит двоичный код на светодиоды, подключенные к портам PORTD (младшие 8 бит) и PORTB (2 старших бита), показанным на рисунке 22.

```
unsigned int temp_res;
void main() {
ADCON1 = 0x80; // Конфигурирова-
ние аналоговых входов и Vref
TRISA = 0xFF; // Все выводы PORTA -
входы
TRISB = 0x3F; // Выводы RB7, RB6 -
выходы
TRISD = 0; // Все выводы PORTD -
выходы
do {
temp_res = Adc_Read(2); // полу-
```

Таблица 3. Описание функции Adc_Read

Прототип	unsigned Adc_Read(unsigned short channel)
Результат	10-битовое беззнаковое число, прочитанное из указанного канала
Описание	Инициализирует внутренний модуль ADC микроконтроллера для работы с тактовым генератором RC. Тактовый генератор определяет длительность преобразования. Параметр channel представляет номер канала, напряжение с которого преобразуется АЦП. Соответствие номеров выводов и номеров каналов приведено в документации на используемый микроконтроллер
Требования	PIC-микроконтроллер со встроенным модулем АЦП. Данные о наличии таких модулей в конкретных устройствах есть в документации (АЦП присутствуют в большинстве представителей семейств P16 и P18). Перед использованием функции необходимо сконфигурировать соответствующий вывод как вход установкой в единичное состояние соответствующего бита регистра TRISx. Также этот вывод должен быть сконфигурирован как аналоговый вход и задан источник опорного напряжения Vref. В ранних версиях mikroC функция может не поддерживаться микроконтроллерами: P18F2331, P18F2431, P18F4331 и P18F4431
Пример	unsigned tmp; ... tmp = Adc_Read(1); /* чтение аналогового значения из канала 1 */

Таблица 4. Описание функции Keypad_Init

Прототип	void Keypad_Init(unsigned *port)
Возвращаемое значение	Нет
Описание	Инициализирует порт для работы с клавиатурой. Должна вызываться перед использованием других функций данной библиотеки
Требования	Нет
Пример	Keypad_Init(&PORTB)

Таблица 5. Описание функции Keypad_Read

Прототип	unsigned short Keypad_Read(void)
Возвращаемое значение	1 – 16 в зависимости от нажатой клавиши и 0, если нет нажатых клавиш
Описание	Проверяет нажатие клавиши. Функция возвращает 1 – 16 в зависимости от нажатой клавиши и 0, если нет нажатых клавиш
Требования	Порт должен быть предварительно проинициализирован с помощью функции Keypad_Init
Пример	kp = Keypad_Read()

Таблица 6. Описание функции Keypad_Released

Прототип	unsigned short Keypad_Released(void)
Возвращаемое значение	Возвращаемое значение 1 – 16 в зависимости от клавиши
Описание	Вызов функции Keypad_Released – блокирующий: функция ждёт, пока клавиша не будет нажата и отпущена. При отпускании функция возвращает код от 1 до 16 в зависимости от клавиши
Требования	Порт должен быть предварительно проинициализирован с помощью функции Keypad_Init
Пример	kp = Keypad_Released()

© СТА-ПРЕСС

```
чить результат преобразования
PORTD = temp_res; // Вывести
младшие 8 битов в PORTD
PORTB = temp_res >> 2; // Вывести
старшие 2 бита на RB7, RB6
} while(1);
}
```

Функции для работы с клавиатурой

Рассмотрим функции для работы с клавиатурой, организованной в виде матрицы 4×4. Данные функции также могут использоваться для обслуживания клавиатур 4×1, 4×2 и 4×3. Схема подключения клавиатуры представлена на рисунке 23.

Для работы с клавиатурой используются библиотечные функции Keypad_Init, Keypad_Read и Keypad_Released. Описание этих функций представлено в таблицах 4, 5 и 6 соответственно.

Следующий программный код может быть использован для проверки клавиатуры. Он поддерживает клавиатуры с матричной организацией от 1 до 4 строк и от 1 до 4 столбцов. Код, возвращаемый библиотечными функциями, перекодируется в ASCII-коды [0..9, A..F]. Дополнительно используется однобайтовый счётчик для вывода общего количества нажатий на клавиши во второй строке ЖКИ.

```
unsigned short kp, cnt;
char txt[5];
void main() {
cnt = 0;
Keypad_Init(&PORTC);
```

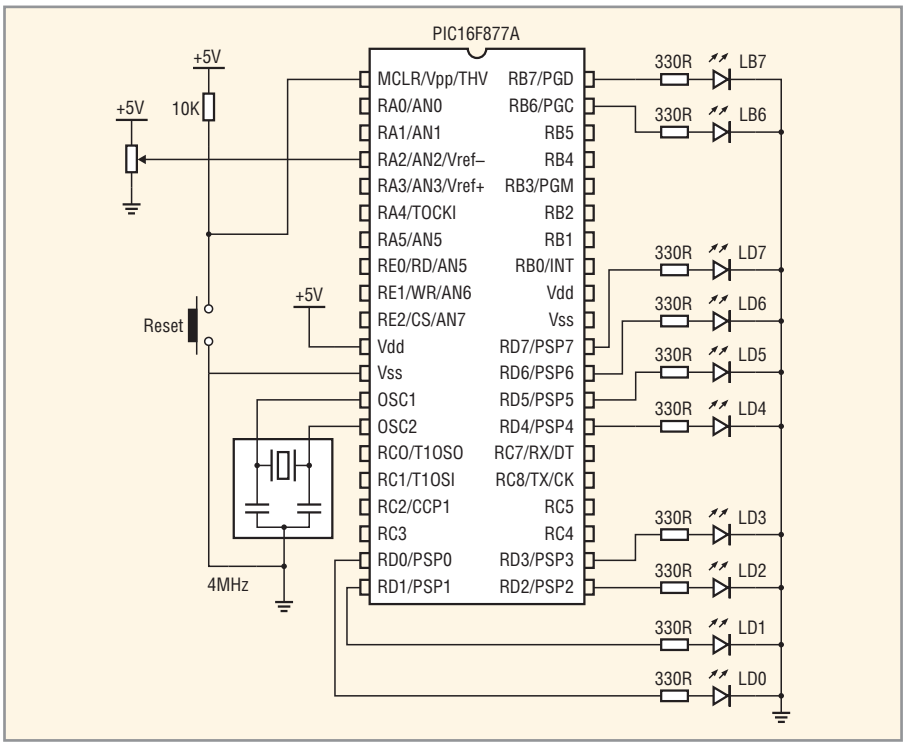

Рис. 22. Схема подключения АЦП

```
Lcd_Init(&PORTB); // Инициализа-
ция LCD на PORTC
Lcd_Cmd(LCD_CLEAR); // Очистка
дисплея
Lcd_Cmd(LCD_CURSOR_OFF); // От-
ключение курсора
Lcd_Out(1, 1, "Key :");
Lcd_Out(2, 1, "Times:");
do {
kp = 0;
//--- Ожидание нажатия клавиши
do
//--- закомментировать неиспольз-
уемую функцию опроса клавиатуры
kp = Keypad_Released();
```

```
//kp = Keypad_Read();
while (!kp);
cnt++;
//--- подготовить значение для
вывода
if (kp > 10)
kp += 54;
else
kp += 47;
//--- вывод его на LCD
Lcd_Chr(1, 10, kp);
WordToStr(cnt, txt);
Lcd_Out(2, 10, txt);
} while (1);
}// конец прорамы
```

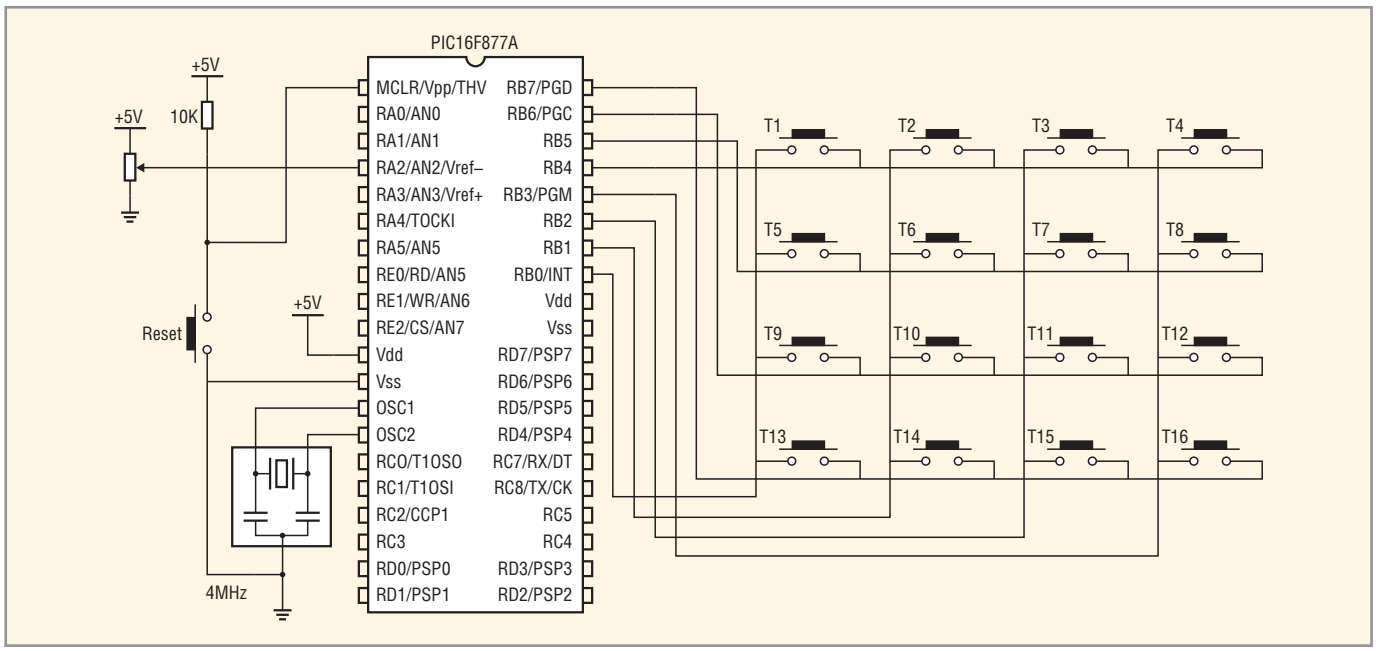

Рис. 23. Схема подключения клавиатуры 4×4

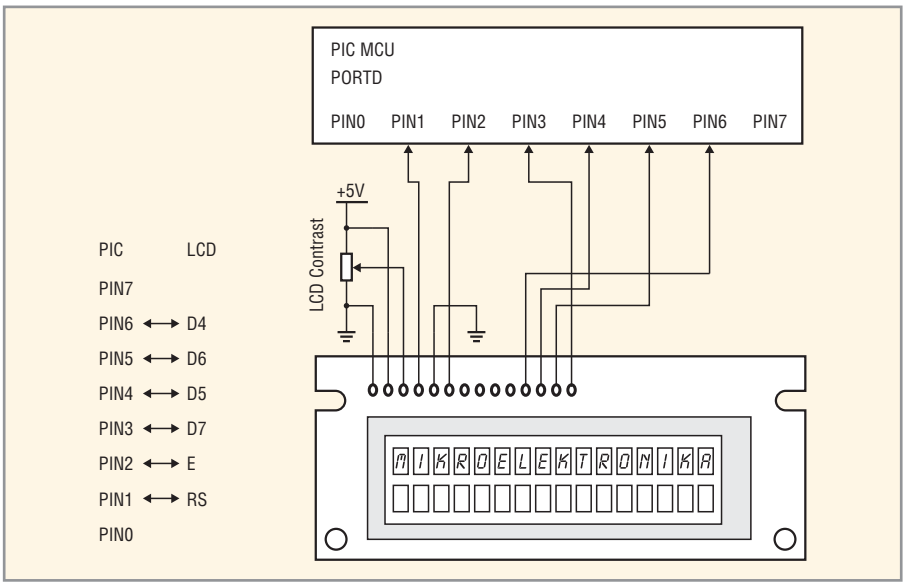

Рис. 24. Схема подключение ЖКИ

Функции для работы с LCD

Рассмотрим функции для работы с наиболее широко распространёнными ЖКИ-контроллерами по четырёхбитному интерфейсу с использованием определённых программистом выводов. Схема подключения контроллера ЖКИ к PIC представлена на рисунке 24.

Перед тем как вызывать библиотечные функции, следует убедиться, что используемый для работы с ЖКИ порт настроен на вывод. Для работы с клавиатурой используются библиотечные функции Lcd_Custom_Config, Lcd_Custom_Out, Lcd_Custom_Out_Cp, Lcd_Custom_Chr, Lcd_Custom_Chr_Cp и Lcd_Custom_Cmd. Описание этих функций представлено в таблицах 7 – 13 соответственно.

Следующий программный код демонстрирует пример использования описанных функций и позволяет отобразить на дисплее строку mikroElektronika.

```
char *text = "mikroElektronika";
void main() {
TRISD = 0; // PORTD на выход
// Инициализировать LCD на порт PORTD
Lcd_Custom_Config(&PORTD,3,5,4,6,
&PORTD,1,0,2);
Lcd_Custom_Cmd(Lcd_CURSOR_OFF);
// Выключить LCD
Lcd_Custom_Out(1, 1, text); //
Вывести текст на LCD
}
```

Таблица 7. Описание функции Lcd_Custom_Config

Прототип	void Lcd_Custom_Config(char * data_port, char D7, char D6, char D5, char D4, char * ctrl_port, char RS, char WR, char EN)
Возвращаемое значение	Нет
Описание	Инициализирует порт данных (data_port) и управления (control_port) в соответствии с заданным назначением разрядов
Требования	Нет
Пример	Lcd_Custom_Config(&PORTD,3,2,1,0,&PORTB,2,3,4)

Таблица 8. Описание функции Lcd_Custom_Out

Прототип	void Lcd_Custom_Out(char row, char col, char *text)
Возвращаемое значение	Нет
Описание	Выводит на LCD text в заданную строку и заданную позицию (аргументы row и col). В качестве text может использоваться как строковая переменная, так и литерал
Требования	Порт, к которому подключен контроллер ЖКИ, должен быть проинициализирован функцией Lcd_Custom_Config
Пример	Вывод «Hello!» на LCD в строку 1, с позиции 3: Lcd_Custom_Out(1, 3, "Hello!");

Таблица 9. Описание функции Lcd_Custom_Out_Cp

Прототип	void Lcd_Custom_Out_Cp(char *text)
Возвращаемое значение	Нет
Описание	Выводит text на ЖКИ, начиная с текущей позиции курсора. В качестве text может использоваться как строковая переменная, так и литерал
Требования	Порт, к которому подключен контроллер ЖКИ, должен быть проинициализирован функцией Lcd_Custom_Config
Пример	Вывод «Here!» с текущей позиции курсора: Lcd_Custom_Out_Cp("Here!")

Таблица 10. Описание функции Lcd_Custom_Chr

Прототип	void Lcd_Custom_Chr(char row, char col, char character)
Возвращаемое значение	Нет
Описание	Вывод символа character на ЖКИ в заданную строку и позицию (аргументы row и col). В качестве character может использоваться как переменная, так и литерал
Требования	Порт, к которому подключен контроллер LCD, должен быть проинициализирован функцией Lcd_Custom_Config
Пример	Вывод «I» на ЖКИ в строку 2, позицию 3: Lcd_Custom_Chr(2, 3, 'i')

Таблица 11. Описание функции Lcd_Custom_Chr_Cp

Прототип	void Lcd_Custom_Chr_Cp(char character)
Возвращаемое значение	Нет
Описание	Выводит символ character на ЖКИ в текущую позицию курсора. В качестве character может использоваться как переменная, так и литерал
Требования	Порт, к которому подключен контроллер ЖКИ, должен быть проинициализирован функцией Lcd_Custom_Config
Пример	Вывод «e» в текущую позицию: Lcd_Custom_Chr_Cp('e')

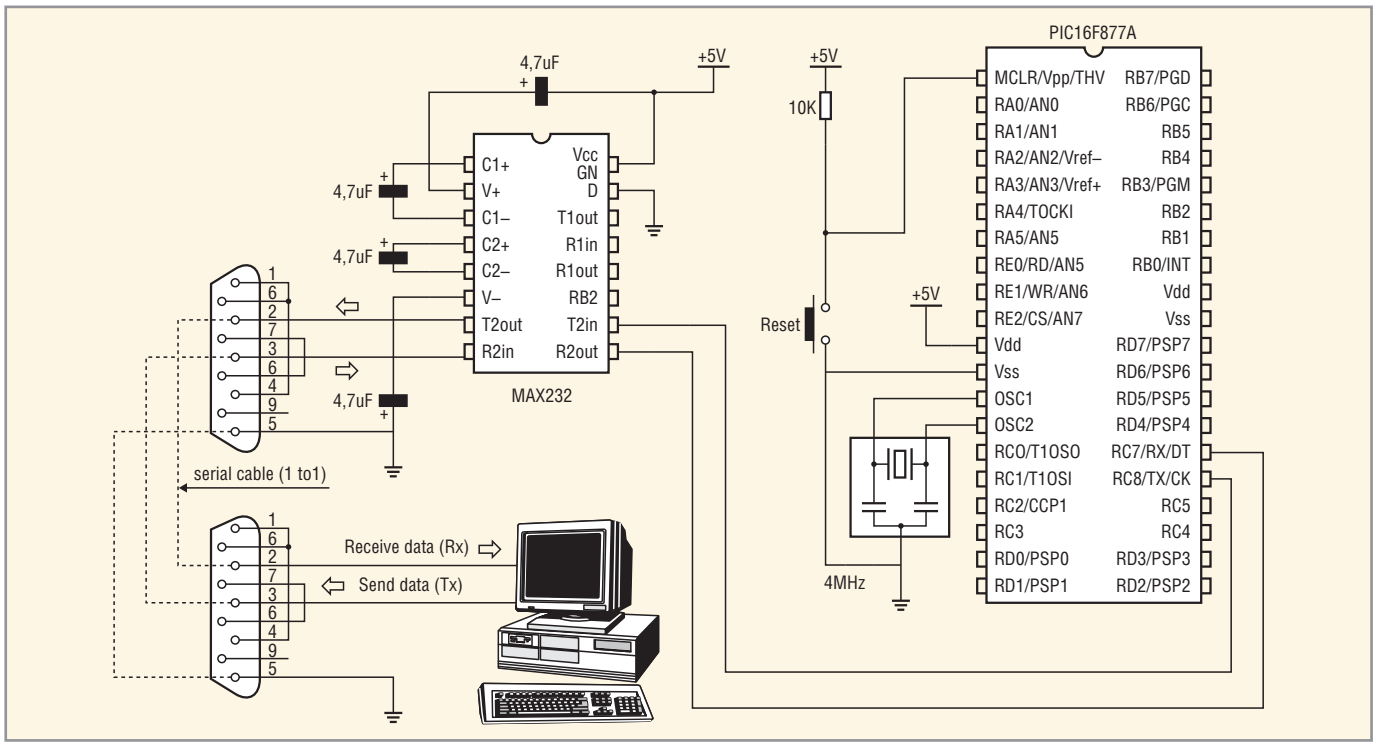

Рис. 25. Схема аппаратного подключения RS-232

Функции для работы с USART

Рассмотрим функции для работы с аппаратной реализацией. Аппаратный модуль USART имеется во многих представителях семейства PIC. Библиотека mikroC для работы с аппаратной реализацией USART предоставляет возможности удобной работы в асинхронном полнодуплексном режиме.

Таблица 12. Описание функции Lcd_Custom_Cmd

Прототип	void Lcd_Custom_Cmd(char command)
Возвращаемое значение	Нет
Описание	Посылает команду command на ЖКИ. Здесь можно использовать одну из предопределённых команд. Список всех команд приведён ниже
Требования	Порт, к которому подключен контроллер ЖКИ, должен быть проинициализирован функцией Lcd_Custom_Config
Пример	Очистка LCD: Lcd_Custom_Cmd(LCD_CLEAR)

Таблица 13. Список команд для управления LCD

Команда LCD	Назначение
LCD_FIRST_ROW	Перемещение курсора в 1-ю строку
LCD_SECOND_ROW	Перемещение курсора во 2-ю строку
LCD_THIRD_ROW	Перемещение курсора в 3-ю строку
LCD_FOURTH_ROW	Перемещение курсора в 4-ю строку
LCD_CLEAR	Стирание дисплея
LCD_RETURN_HOME	Возврат курсора в исходное положение и возврат сдвинутого изображения дисплея в первоначальное состояние. Данные в памяти контроллера дисплея не затрагиваются
LCD_CURSOR_OFF	Выключение курсора
LCD_UNDERLINE_ON	Включение курсора «подчёркивание»
LCD_BLINK_CURSOR_ON	Включение мерцания курсора
LCD_MOVE_CURSOR_LEFT	Сдвиг курсора влево без изменения данных в памяти контроллера дисплея
LCD_MOVE_CURSOR_RIGHT	Сдвиг курсора вправо без изменения данных в памяти контроллера дисплея
LCD_TURN_ON	Включение дисплея
LCD_TURN_OFF	Выключение дисплея
LCD_SHIFT_LEFT	Сдвиг экрана дисплея влево без изменения памяти контроллера дисплея
LCD_SHIFT_RIGHT	Сдвиг экрана дисплея вправо без изменения памяти контроллера дисплея

Таблица 14. Описание функции Usart_Init

Прототип	void Usart_Init(const unsigned long baud_rate)
Возвращаемое значение	Нет
Описание	Инициализация аппаратуры модуля USART требуемой скоростью обмена. Для подробной информации о разрешённых скоростях обмена при заданных частотах тактового генератора следует обратиться к документации на микроконтроллер. Если будет задана запрещённая скорость обмена, компилятор сообщит об ошибке
Требования	Необходим микроконтроллер с аппаратным USART. Функция Usart_Init должна быть вызвана перед использованием всех остальных функций библиотеки работы с USART
Пример	Инициализация аппаратуры USART и установка скорости обмена 2400 bps: Usart_Init(2400)

Можно легко организовать связь с другими устройствами, поддерживающими протокол RS-232, например, с персональным компьютером (см. рис. 25). Для этого потребуется микроконтроллер с аппаратным USART, например PIC16F877. В программе можно использовать все функции, приведённые ниже.

Для работы с USART используются библиотечные функции Usart_Init, Usart_Data_Ready, Usart_Read и Usart_Write. Описание этих функций представлено в таблицах 14 – 17 соответственно.

Микроконтроллеры, имеющие по два модуля USART, например P18F8520,

Таблица 15. Описание функции Usart_Data_Ready

Прототип	unsigned short Usart_Data_Ready(void)
Возвращаемое значение	Функция возвращает значение 1, если есть принятые данные, и 0 в противном случае
Описание	Эту функцию следует использовать для проверки наличия данных для чтения в приёмном регистре
Требования	Аппаратный модуль USART должен быть предварительно проинициализирован и задана скорость обмена с помощью функции Usart_Init
Пример	Если данные готовы, прочитать их: int receive; ... if (Usart_Data_Ready()) receive = Usart_Read;

Таблица 16. Описание функции Usart_Read

Прототип	unsigned short Usart_Read(void)
Возвращаемое значение	Возвращает значение принятого байта. Если байт не принят, возвращает 0
Описание	Функция считывает принятые данные из приёмного регистра аппаратуры USART. Для проверки наличия принятых данных следует использовать функцию Usart_Data_Ready
Требования	Аппаратный модуль USART должен быть предварительно проинициализирован, а скорость обмена задана с помощью функции Usart_Init
Пример	Если данные готовы, прочитать их: int receive; ... if (Usart_Data_Ready()) receive = Usart_Read();

Таблица 17. Описание функции Usart_Write

Прототип	void Usart_Write(unsigned short data)
Возвращаемое значение	Нет
Описание	Функция передаёт байт (data) по USART
Требования	Аппаратный модуль USART должен быть предварительно проинициализирован, а скорость обмена задана с помощью функции Usart_Init
Пример	int chunk = 0x1E; Usart_Write(chunk); /* послать chunk по USART */

Таблица 18. Описание функции Hid_Enable

Прототип	void Hid_Enable(unsigned *readbuff, unsigned *writebuff)
Возвращаемое значение	Нет
Описание	Разрешает USB HID-обмен. Аргументы readbuff и writebuff – указатели на буферы чтения и записи соответственно, используемые для обмена
Требования	Эта функция должна быть вызвана перед использованием любых других из библиотеки USB HID
Пример	Hid_Enable(&rd, &wr)

Таблица 19. Описание функции Hid_Read

Прототип	unsigned short Hid_Read(void)
Возвращаемое значение	Количество символов в буфере чтения, принятое от хоста
Описание	Принимает сообщение от хоста и сохраняет его в буфере чтения. Функция возвращает количество символов в буфере чтения
Требования	Обмен с USB HID-устройством должен быть предварительно разрешён с помощью функции Hid_Enable
Пример	get = Hid_Read()

Таблица 20. Описание функции Hid_Write

Прототип	unsigned short Hid_Write(unsigned *writebuff, unsigned short len)
Возвращаемое значение	1, если данные успешно переданы, 0 в противном случае.
Описание	Функция отправляет данные из буфера записи writebuff хосту. Буфер записи – это тот же аргумент, что использовался при инициализации обмена функцией Hid_Enable. Аргумент len определяет количество байтов для передачи
Требования	Вызов функции следует повторять до тех пор, пока данные не будут успешно отправлены. Обмен с USB HID-устройством должен быть предварительно разрешён с помощью функции Hid_Enable

Пример	// повторять, пока данные не отправятся
	while(!Hid_Write(&my_Usb_Buff, 1));

Таблица 21. Описание функции Hid_Disable

Прототип	void Hid_Disable(void)
Возвращаемое значение	Нет
Описание	Запрещает обмен с USB HID-устройством.
Требования	Обмен с USB HID-устройством должен быть предварительно разрешён с помощью функции Hid_Enable
Пример	Hid_Disable()

требуют задать модуль, который будет использоваться. Для этого достаточно просто приписать цифру 1 или 2 к названию функции. Например, Usart_Write2(). Также, с целью обратной совместимости с предыдущими версиями компилятора и облегчения управления кодами, МК с несколькими модулями USART имеют Usart-библиотеку, которая идентична Usart1 (т.е. можно использовать Usart_Init() вместо Usart1_Init() для операций с USART).

Следующий пример программы демонстрирует простой обмен данными по USART. Когда МК принимает данные, он немедленно отправляет те же данные обратно. Если подключить PIC к ПК (см. рис. 25), можно проверить обмен с терминалом для связи по RS-232. В среде разработки терминал запускается из выпадающего меню командой Tools → Terminal.

```
unsigned short i;
void main() {
// Инициализация модуля USART (8
бит, скорость обмена 2400 бод,
без к/ч)
Usart_Init(2400);

do {
if (Usart_Data_Ready()) { // Если
данные приняты
i = Usart_Read(); // Прочитать,
что принято
Usart_Write(i); // и отправить
назад по USART
}
} while (1);
} // end
```

Функции для работы с USB HID-устройствами

Универсальная последовательная шина USB (Universal Serial Bus) представляет собой стандарт на последовательный интерфейс обмена для подключения к компьютеру широкого диапазона устройств, таких как сотовые телефоны, игровые приставки, PDA, и т.п. Среда MikroC включает в себя функции для работы с HID (Human Interface Devices) – устройствами интерфейса человека с машиной через USB. HID-устройства – это разновидность компьютерных устройств, напрямую взаимодействующих с человеком, например, клавиатура, мышь, графический планшет и т.п.

Каждый проект на базе библиотеки USB HID должен включать в себя исходный файл дескриптора, содержащий идентификатор, и название производителя, идентификатор и название продукта, размер посылки и другую важную информацию. Для создания файла дескриптора следует использовать USB HID-терминал, интегрированный в mikroC и вызываемый командой Tools → USB HID Terminal. Название файла дескриптора по умолчанию USBdsc.c, но его можно изменить.

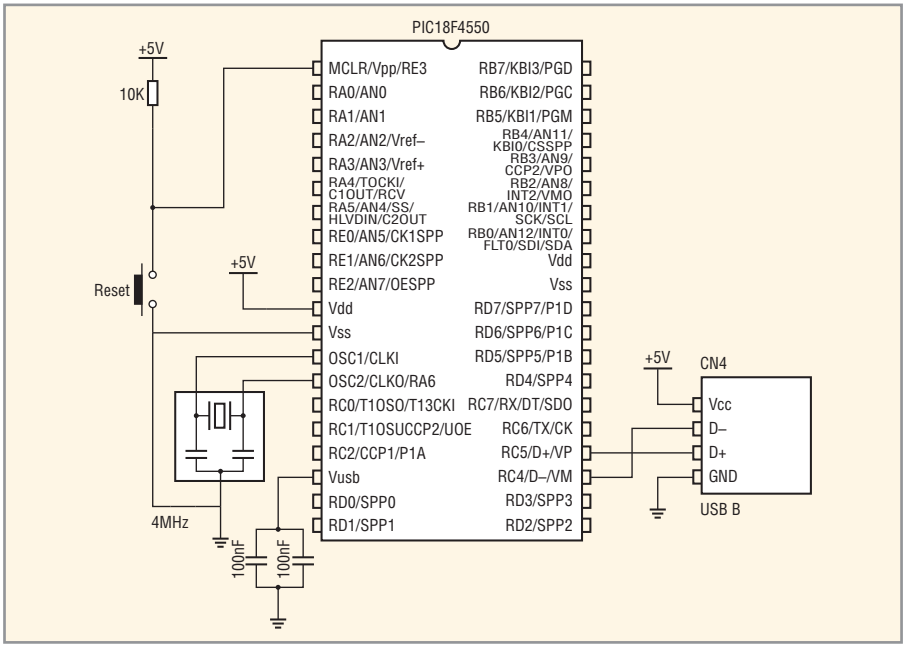

Рис. 26. Схема подключения USB-интерфейса к микроконтроллеру

Код программы, находящийся в каталоге с примерами, работает на частоте 48 МГц. Флаги в этом коде не следует изменять, предварительно не ознакомившись с соответствующей документацией. Для работы с USB используются библиотечные функции Hid_Enable, Hid_Read, Hid_Write и Hid_Disable. Описание этих функций представлено в таблицах 18 – 21 соответственно.

Следующий пример (см. дополнительные материалы на сайте журнала) непрерывно отправляет последовательность чисел 0 – 255 в компьютер через USB. Схема подключения USB-интерфейса к микроконтроллеру показана на рисунке 26. Файл usbdsc.c должен быть включен в проект средствами среды mikroC IDE или директивой #include в исходном коде.

Продолжение следует