Руководство по программированию USB модуля WoodmanUSB

WUSB API Version 2.01

Список функций для работы с модулем WoodmanUSB:

WUSB_Open	Открытие доступа к модулю
WUSB_SetupPortA	Установка режима работы порта PORTA
WUSB_SetupPortB	Установка режима работы порта PORTB
WUSB_WritePortA	Функция записи данных в порт PORTA
WUSB_ReadPortA	Функция чтения данных из порта PORTA
WUSB_WritePortB	Функция записи данных в порт PORTB
WUSB_ReadPortB	Функция чтения данных из порта PORTB
WUSB_RawWritePortB	Запись данных в порт PORTB с дополнительными атрибутами
WUSB_RawReadPortB	Чтение данных из порта PORTB с дополнительными атрибутами
WUSB_GetFIFOStatus	Определение статуса FIFO буферов модуля
WUSB_SetTimeOuts	Установка значений тайм-аутов для операций чтения и записи
WUSB_GetTimeOuts	Получение значений тайм-аутов для операций чтения и записи
WUSB_AbortWritePortB	Принудительная остановка процесса записи данных в порт PORTB
WUSB_AbortReadPortB	Принудительная остановка процесса чтения данных из порта PORTB
WUSB_Close	Закрытие соединения с модулем

WUSB_Open

Функция открывает доступ к модулю WoodmanUSB.

*int **WUSB_Open**();*

Входные параметры:

отсутствуют

Возвращаемое значение:

WUSB_OK в случае успешного открытия устройства, иначе WUSB_ERROR

Пример использования:

```
int status = WUSB_Open();
if(status == WUSB_OK)
{
   //устройство открыто успешно
}
else
{
   //устройство открыть не удалось
}
```

WUSB_SetupPortA

С помощью этой функции можно задать направление ввода/вывода для порта PORTA.

*int **WUSB_SetupPortA**(unsigned char iomsk);*

Входные параметры:

iomsk

Маска ввода/вывода. Если соответствующий бит этой маски равен 1 – то соответствующий бит порта PORTA настраивается на вывод (относитльно модуля), если же бит маски равен 0 – то соответствующий бит порта настраивается на ввод информации.

Возвращаемое значение:

WUSB_OK в случае успешной установки, иначе WUSB_ERROR

Пример использования:

```
WUSB_SetupPortA(255); // все биты порта PORTA настроить на вывод
WUSB_SetupPortA(0);   // все биты порта PORTA настроить на ввод
WUSB_SetupPortA(14);  // 2,4 биты порта на вывод, остальные на ввод
```

WUSB_WritePortA

Записывает один байт данных в порт PORTA.

*int **WUSB_WritePortA**(unsigned char value);*

Входные параметры:

value

байт данных, который необходимо записать в порт.

Возвращаемое значение:

WUSB_OK в случае успешной записи данных, иначе WUSB_ERROR

Пример использования:

```
unsigned char data = 30;
int status = WUSB_WritePortA(data);
if(status == WUSB_ERROR)
{
   //ошибка записи данных
}
```

WUSB_ReadPortA

Читает один байт данных из порта PORTA.

*int **WUSB_ReadPortA**(unsigned char* value);*

Входные параметры:

value

указатель на переменную, в которую будет помещен результат чтения порта PORTA.

Возвращаемое значение:

WUSB_OK в случае успеха чтения данных, иначе WUSB_ERROR

Пример использования:

```
unsigned char data;
int status = WUSB_ReadPortA(&data);
if(status == WUSB_ERROR)
{
   //ошибка чтения данных
}
```

WUSB_SetupPortB

Функция устанавливает режим работы порта PORTB.

*int **WUSB_SetupPortB**(unsigned char param);*

Входные параметры:

param

одна из перечисленных ниже констант, определяющих устанавливаемый режим работы.

ASYNC_MODE	- асинхронный режим
SYNC_MODE_EXTERNAL_CLK	- синхронный режим с внешним тактированием
SYNC_MODE_INTERNAL_CLK_30MHZ	- синхронный режим с внутренним тактированием 30 МГц
SYNC_MODE_INTERNAL_CLK_48MHZ	- синхронный режим с внутренним тактированием 48 МГц

Возвращаемое значение:

WUSB_OK в случае успешной установки, иначе WUSB_ERROR

Пример использования:

```
int status = WUSB_SetupPortB(SYNC_MODE_INTERNAL_CLK_48MHZ);
if(status == WUSB_ERROR)
{
   //ошибка установки
}
```

WUSB_WritePortB

Функция записывает массив данных размером *bufsize* байт из буфера *buf* в порт PORTB.

*int **WUSB_WritePortB**(char *buf, unsigned int bufsize, unsigned int* dwWrite);*

Входные параметры:

buf

адрес на буфер, в котором находятся данные для отправки в порт

bufsize

колличество байт, которые необходимо записать

dwWrite

указатель на переменную, в которую будет помещено число действительно записанных байт в порт

Возвращаемое значение:

WUSB_OK – в случае успеха записи;
WUSB_ERROR – в случае ошибки записи;
WUSB_TIMEOUT – в случае истечения времени тайм-аута.

Замечания:

1. Размер буфера, который можно записать в порт PORTB используя данную функцию, может иметь величину от 1 байта до 65535 байт включительно.

2. Функция заканчивает свою работу, чтолько после того как все из *bufsize* байт будут переданы в модуль. В том случае, если по какой-либо причине внешнее устройство, считывающие данные через порт PORTB, не делает этого, функция 'подвесит' поток в рамках которого она выполняется. Поэтому рекомендуется операцию записи данных осуществлять не в основном рабочем потоке. В связи с этим в функции предусмотрено использование тайм-аутов. Т.е. по истечении определенного времени, если операция записи не завершена до конца, автоматически происходит принудительное завершение работы функции. В качестве результата возвращается код WUSB_TIMEOUT. По умолчанию, значение времени тайм-аута для функции *WUSB_WritePortB*() равно 1000 мс. Подробнее о тайм-аутах см. функции *WUSB_SetTimeOuts()* и *WUSB_GetTimeOuts().*

Пример использования:

```
char buf[4096];
unsigned int dwWrite = 0;
int status = WUSB_WritePortB(buf, sizeof(buf), &dwWrite);
if(status == WUSB_ERROR)
{
   //ошибка записи
}
```

```
else if(status == WUSB_TIMEOUT)
{
   //истечение времени тайм-аута
}
```

WUSB_ReadPortB

Функция читает *bufsize* байт из порта PORTB в буфер *buf*.

*int **WUSB_ReadPortB**(char *buf, unsigned int bufsize, unsigned int* dwRead);*

Входные параметры:

buf

адрес на буфер, в который необходимо поместить прочтенные данные

bufsize

колличество байт, которые необходимо прочитать.

dwRead

указатель на переменную, в которую будет помещено число действительно прочтенных байт из порта

Возвращаемое значение:

WUSB_OK – в случае успеха чтения;

WUSB_ERROR – в случае ошибки чтения;

WUSB_TIMEOUT – в случае истечения времени тайм-аута.

Замечания:

1. Количество считаемых байт данных из порта PORTB модуля за одно обращение к функции чтения жестко фиксировано. Объем считываемых данных всегда должен быть кратен 512 но при этом не превышать величины 65024 (ограничение драйвера) байт, т.е. может принимать значения 512, 1024, 2048, ..., 65024 байт. В действительности, возможно чтение меньшего объема данных, чем было указано в аргументах функции. Для принудительной отсылки пакета данных произвольной длинны необходимо использовать вывод модуля PKTEND.

2. Функция заканчивает свою работу, только после того как все из *bufsize* байт будут прочтены. В том случае, если по какой-либо причине внешнее устройство, записывающее данные в порт PORTB не делает этого, функция 'подвесит' поток в рамках которого она выполняется. Поэтому рекомендуется операцию чтения данных осуществлять не в основном рабочем потоке. В связи с этим в функции предусмотрено использование тайм-аутов. Т.е. по истечении определенного времени, если операция чтения не завершена до конца, автоматически происходит принудительное завершение работы функции. В качестве результата возвращается код WUSB_TIMEOUT. По умолчанию, значение времени тайм-аута для функции *WUSB_ReadPortB*() равно 1000 мс. Подробнее о тайм-аутах см. функции *WUSB_SetTime Outs()* и *WUSB_GetTimeOuts().* Для принудительного завершения записи данных (даже если необходимо передать объем данных меньший чем размер конечной точки) может быть использован вывод модуля PKTEND.

Пример использования:

```
char buf[4096];
unsigned int dwRead = 0;
int status = WUSB_ReadPortB(buf, sizeof(buf), &dwRead);
if(status == WUSB_ERROR)
{
   //ошибка чтения
}
else if(status == WUSB_TIMEOUT)
{
   //истечение времени тайм-аута
}
```

WUSB_RawWritePortB

Функция записывает массив данных размером *bufsize* байт из буфера *buf* в порт PORTB

*int **WUSB_RawWritePortB**(char *buf, unsigned int bufsize, unsigned int* dwWrite);*

Замечания:

Данная функция по своему интерфейсу полностью аналогична функции *WUSB_WritePortB()*. Ее отличие состоит в том, что в ней, во-первых, отсутствует блок автоматической обработки тайм-аутов. Во вторых, отсутствует блок слежения за запросами к драйверу. Последнее означает, что если в процессе работы этой функции из другого программного потока будет совершена попытка обращения к драйверу, последний может зависнуть. В остальных функциях блок слежения приостанавливает новый запрос до тех пор, пока не завершится обработка предыдущего. Подобные упрощения снижают безопасность работы данной функции, однако позволяют увеличить скорость передачи данных на 10% по сравнению с функцией *WUSB_WritePortB()*. Поскольку автоматической обработки тайм-аутов в функции нет, в случае необходимости для принудительного завершения работы необходимо использовать функцию *WUSB_AbortWritePortB()*.

WUSB_RawReadPortB

Функция читает *bufsize* байт из порта PORTB в буфер *buf*.

*int **WUSB_RawReadPortB**(char *buf, unsigned int bufsize, unsigned int* dwRead);*

Замечания:

Данная функция по своему интерфейсу полностью аналогична функции *WUSB_ReadPortB()*. Ее отличие состоит в том, что в ней, во-первых, отсутствует блок автоматической обработки тайм-аутов. Во вторых, отсутствует блок слежения за запросами к драйверу. Последнее означает, что если в процессе работы этой функции из другого программного потока будет совершена попытка обращения к драйверу, последний может зависнуть. В остальных функциях блок слежения приостанавливает новый запрос до тех пор, пока не завершится обработка предыдущего. Подобные упрощения снижают безопасность работы данной функции, однако позволяют увеличить скорость передачи данных на 10% по сравнению с функцией *WUSB_ReadPortB()*. Поскольку автоматической обработки тайм-аутов в функции нет, в случае необходимости для принудительного завершения работы необходимо использовать функцию *WUSB_AbortReadPortB()*.

WUSB_GetFIFOStatus

Функция возвращает состояние FIFO буферов модуля.

*int **WUSB_GetFIFOStatus**(unsigned char* INstatus, unsigned char* OUTstatus);*

Входные параметры:

INstatus

указатель на переменную, в которую будет помещено значение состояния входного FIFO буфера (IN_FIFO);

OUTstatus

указатель на переменную, в которую будет помещено значение состояния выходного FIFO буфера (OUT_FIFO);

Значения констант состояния FIFO буфера:

FIFO_EMPTY	– буфер пуст, в нем содержится 0 байт данных;
FIFO_FULL	– буфер полный, нем содержится 1024 байт данных (1024 – размер FIFO буфера для модуля WoodmanUSB);
FIFO_NOTEMPTY	– в буфере содержится > 0 но < 1024 байт данных.

Возвращаемое значение:

WUSB_OK в случае успеха операции, иначе WUSB_ERROR

Пример использования:

```
unsigned char Instatus, OUTstatus;
int status = WUSB_GetFIFOStatus(&INstatus, &OUTstatus);
if(status == WUSB_ERROR)
{
   //ошибка
```

```
}
```

WUSB_SetTimeOuts

С помощью этой функции можно установить значения тайм-аута для операций чтения и записи данных в порт POPRTB.

*int **WUSB_SetTimeOuts**(unsigned int ReadTimeOut, unsigned int WriteTimeOut);*

Входные параметры:

ReadTimeOut

значение тайм-аута для операции чтения данных из порта PORTB в миллисекундах;

WriteTimeOut

значение тайм-аута для операции записи данных в порт PORTB в миллисекундах.

Возвращаемое значение:

WUSB_OK в случае успеха операции, иначе WUSB_ERROR

Пример использования:

```
// устанавливаем значение тайм-аута для обоих операций в 2 сек.
unsigned int ReadTimeOut  = 2000;
unsigned int WriteTimeOut = 2000;

int status = WUSB_SetTimeOuts(ReadTimeOut, WriteTimeOut);
if(status == WUSB_ERROR)
{
   //ошибка установки
}
```

WUSB_GetTimeOuts

Функция возвращает значения тайм-аута для операций чтения и записи данных в порт POPRTB.

*int **WUSB_GetTimeOuts**(unsigned int* ReadTimeOut, unsigned int* WriteTimeOut);*

Входные параметры:

ReadTimeOut

указатель на переменную, в которую будет помещено значение тайм-аута для операции чтения данных из порта PORTB в миллисекундах;

WriteTimeOut

указатель на переменную, в которую будет помещено значение тайм-аута для операции записи данных в порт PORTB в миллисекундах.

Возвращаемое значение:

WUSB_OK в случае успеха операции, иначе WUSB_ERROR

Замечания:

По умолчанию, значения тайм-аутов для обоих операций составляет 1000 мс.

Пример использования:

```
unsigned int ReadTimeOut, WriteTimeOut;

int status = WUSB_GetTimeOuts(&ReadTimeOut, &WriteTimeOut);
if(status == WUSB_ERROR)
{
   //ошибка установки
}
```

WUSB_AbortWritePortB

Функция принудительно завершает начатый процесс записи данных в порт PORTB.

*bool **WUSB_AbortWritePortB**();*

Входные параметры:

отсутствуют

Возвращаемое значение:

true в случае успеха операции, иначе *false*.

Замечания:

В случае применения этой функции, функции *WUSB_WritePortB()* и *WUSB_RawWritePortB()* в качестве значения числа записанных байт данных вернет 0, хотя реально это не так. В модуль будут переданы те данные, которые были отправлены до вызова функции *WUSB_AbortWritePortB()*.

Пример использования:

```
bool b = WUSB_AbortWritePortB();
if(b == false)
{
   //завершить запись не удалось
}
```

WUSB_AbortReadPortB

Функция принудительно завершает начатый процесс чтения данных из порта PORTB.

*bool **WUSB_AbortReadPortB**();*

Входные параметры:

отсутствуют

Возвращаемое значение:

true в случае успеха операции, иначе *false*.

Замечания:

В случае применения этой функции, функции *WUSB_ReadPortB()* и *WUSB_RawReadPortB()* в качестве значения числа прочтенных байт данных вернет 0. Буфер для прочтенных данных так же будет пуст, даже если до вызова функции *WUSB_AbortReadPortB()* модуль успел передать часть данных.

Пример использования:

```
bool b = WUSB_AbortReadPortB();
if(b == false)
{
   //завершить чтение не удалось
}
```

WUSB_Close

Функция закрытия соединения с модулем

*int **WUSB_Close**();*

Входные параметры:

отсутствуют

Возвращаемое значение:

WUSB_OK в случае успеха операции, иначе WUSB_ERROR

Пример использования:

```
int status = WUSB_Close();
if(status == WUSB_ERROR)
{
   //ошибка закрытия соединения
}
```

KERNELCHIP

Россия, Москва

Телефон: +7 (917) 516 99 51

http://www.kernelchip.ru

port@kernelchip.ru