

Описание библиотеки
функций
для плат серии CAN-200

Руководство пользователя.
v1.0

Операционная система: Windows 98/ME/2000/XP

Оглавление

Назначение	3
Условия работы	3
Введение	4
Функции инициализации платы и драйверной библиотеки	5
CANOpen	5
CANClose	5
SelectCAN	6
chBaseAddress	6
HardReset	7
GetConfig	7
Описание структуры TCAN_VPDDData	8
Функции для режимов BasicCAN и PeliCAN	9
SetWorkMode	9
GetWorkMode	9
SetDriverMode	10
SetCANSpeed	10
SetInterruptSource	11
GetStatus	12
SetTxBuffer	13
GetRxBuffer	14
SetCommand	15
GetEventData	16
Описание структуры RX_TX_Buffer	17
Функции для режима BasicCAN	18
B_SetInputFilter	18
Функции для режима PeliCAN	19
P_SetRxErrorCounter	19
P_SetTxErrorCounter	20
P_GetTxErrorCounter	20
P_SetErrorWarningLimit	21
P_GetErrorWarningLimit	21
P_GetArbitrationLostCapture	22
P_GetRxMessageCounter	23
P_GetErrorCode	24
P_SetInputFilter	26
Описание структуры TEventData	27
Пример обработки прерываний	28

Назначение

Библиотека функций предназначена для предоставления доступа к ресурсам плат серии CAN-200.

Условия работы

Установленное оборудование:

IBM PC совместимый ПК 80386 и выше

Программное обеспечение:

Microsoft Visual C++ версии 6.0

Поддерживаемые операционные системы:

- Windows 98/ME/2000/XP

Поддерживаемые платы:

- CAN-200PC
- CAN-200MP
- CAN-200PC104
- CAN-200PCI

Примечание:

1. Режим PeliCAN поддерживается платами серии CAN-200 с CAN-контроллером SJA1000. Платами серии CAN-200 с CAN-контроллерами PCA82C200 поддерживается только режим BasicCAN.

Введение

Платы серии CAN-200 содержат два (CAN-200PC, CAN-200MP, CAN-200PC104, CAN-200PCI) или четыре (CAN-200-6U) полностью независимых CAN-канала. Каждый CAN-канал построен на CAN-контроллере PCA82C200 фирмы Philips. Этот CAN-контроллер поддерживает на аппаратном уровне обмен данными по протоколу CAN 2.0 part A. В 2002 году CAN-контроллер PCA82C200 снят с производства и в качестве его замены на платы серии CAN-200 устанавливается CAN-контроллер SJA1000, поддерживающий (в режиме PeliCAN) обмен данными по протоколу CAN 2.0 part B active. Этот CAN-контроллер имеет два режима функционирования: BasicCAN и PeliCAN. Режим BasicCAN является режимом, обеспечивающим совместимость с CAN-контроллером PCA82C200. В этом режиме возможен обмен только стандартными кадрами (с 11-битным идентификатором). Режим PeliCAN является режимом, обеспечивающим более гибкую настройку CAN-контроллера, удобный мониторинг CAN-сети и предоставляет возможность обмена кадрами с расширенным идентификатором (29-битным).

Функции инициализации платы и драйверной библиотеки

CANOpen

Описание:

Получение доступа к драйверу устройства.

Прототип:

unsigned short CANOpen(int)

Входные параметры:

Порядковый номер устройства CAN-200 в системе

Возвращаемое значение:

0 - доступ к драйверу устройства получен

1- ошибка открытия драйвера (устройство с таким номером в системе отсутствует)

Включаемые файлы:

#include "wdmcan.h"

Примечание:

Нумерация устройств начинается с нуля.

CANClose

Описание:

Закрытие доступа к драйверу устройства.

Прототип:

void CANClose(int)

Входные параметры:

Порядковый номер устройства CAN-200 в системе

Возвращаемое значение:

нет

Включаемые файлы:

#include "wdmcan.h"

Примечание:

Нумерация устройств начинается с нуля.

SelectCAN

Описание:

Выбор номера устройства CAN-200

Прототип:

int SelectCAN (int)

Входные параметры:

Порядковый номер устройства CAN-200 в системе.

Возвращаемое значение:

0 – выбор устройства произведен

1 – ошибка при выборе устройства (доступ к драйверу устройства не был открыт)

Включаемые файлы:

#include "wdmcan.h"

Примечание:

Для удобства работы с несколькими платами CAN-200 рекомендуется выполнить открытие драйвера (функция CANOpen) для всех установленных плат, управление которыми предполагается выполнять в текущем процессе. Данной функцией производится выбор устройства (платы) для управления. Нумерация устройств начинается с нуля.

chBaseAddress

Описание:

Установка канала платы, с которым будут работать все остальные функции драйверной библиотеки.

Прототип:

void chBaseAddress(unsigned int)

Входные параметры:

1 – установить в качестве базового базовый адрес первого канала

2 – установить в качестве базового базовый адрес второго канала

Возвращаемое значение:

нет

Включаемые файлы:

#include "wdmcan.h"

Примечание:

После вызова этой функции все остальные функции библиотеки будут работать с CAN-каналом по новому базовому адресу.

HardReset

Описание:

Производит аппаратный сброс контроллера канала.

Прототип:

`void HardReset(void)`

Входные параметры:

нет

Возвращаемое значение:

нет

Включаемые файлы:

`#include "wdmcan.h"`

Примечание:

Функция не работает с платами CAN-200PCI.

GetConfig

Описание:

Получение параметров устройства CAN-200

Прототип:

`unsigned short GetConfig(pTCAN_VPDData);`

Входные параметры:

указатель на структуру TCAN_VPDData, в которую будут записаны параметры устройства.

Возвращаемое значение:

Всегда возвращает - 0

Включаемые файлы:

`#include "wdmcan.h"`

Примечание:

1. Функция получает параметры текущего устройства, выбранного функцией SelectCAN.
2. Описание структуры TCAN_VPDData приведено после описания функций.

Описание структуры TCAN_VPDData

Описание:

Структура, используемая для получения информации о конфигурации платы (устройства), выбранной функцией SelectCAN.

Прототип:

```
typedef struct
{
    char szName[128];
    char szSN[10];
    USHORT wPorts1;
    USHORT wPorts2;
    USHORT wIRQ;
} TCAN_VPDData, *pTCAN_VPDData;
```

Описание полей структуры:

szName[128] – название платы (устройства).

Для плат CAN-200PC, CAN-200MP, CAN-200PC104 возвращается следующая строка: CAN-200PC/MP/PC104

Для плат CAN-200PCI возвращается следующая строка: CAN-200PCI vX.Y

, где vX.Y – номер версии платы.

szSN[10] – серийный номер платы. Возвращается только для плат CAN-200PCI

wPorts1 – базовый адрес первого канала платы

wPorts2 – базовый адрес второго канала платы

wIRQ – номер вектора прерывания, используемый платой

Включаемые файлы:

```
#include "wdmcan.h"
```

Примечание:

нет

Функции для режимов BasicCAN и PeliCAN

SetWorkMode

Описание:

Изменить режим работы CAN-контроллера.

Прототип:

unsigned short SetWorkMode(unsigned short)

Входные параметры:

BasicCAN - контроллер переводится в режим BasicCAN

PeliCAN - контроллер переводится в режим PeliCAN

Возвращаемое значение:

0 функция успешно завершена

1 некорректный входной параметр

Включаемые файлы:

#include "wdmcan.h"

Примечание:

Функция не работает с CAN-контроллерами PCA82C200. CAN-контроллер PCA82C200 всегда находится в режиме BasicCAN.

GetWorkMode

Описание:

Получить режим работы контроллера канала.

Прототип:

unsigned short GetWorkMode(void)

Входные параметры:

нет

Возвращаемое значение:

BasicCAN - контроллер находится в режиме BasicCAN

PeliCAN - контроллер находится в режиме PeliCAN

Включаемые файлы:

#include "wdmcan.h"

Примечание:

CAN-контроллер PCA82C200 всегда находится в режиме BasicCAN.

SetDriverMode

Описание:

Функция настройки выходного формирователя CAN-контроллера.

Прототип:

void SetDriverMode (unsigned short)

Входные параметры:

Параметр должен принимать значение 1Bh

Возвращаемое значение:

нет

Включаемые файлы:

#include "wdmcan.h"

Примечание:

Функцию необходимо вызвать один раз при инициализации CAN-контроллера после включения питания. Без вызова этой функции прием/передача данных CAN-контроллером будет невозможна!

SetCANSpeed

Описание:

Установка скорости обмена по шине CAN в кБит/с.

Прототип:

unsigned short SetCANSpeed(unsigned int)

Входные параметры:

Параметр должен принимать одно из следующих значений:
1000, 800, 500, 250, 125, 50, 20, 10

Возвращаемое значение:

0	-	функция успешно завершена
1	-	некорректный входной параметр

Включаемые файлы:

#include "wdmcan.h"

Примечание:

нет

SetInterruptSource

Описание:

Разрешение источников прерывания.

Прототип:

```
void SetInterruptSource(unsigned short)
```

Входные параметры:

Байт маски прерывания.

Если бит №х в параметре mask равен лог.1, то прерывание по причине, описанной в байте идентификации прерывания (см. функцию GetInterruptSouce) в бите №х будет разрешено.

Возвращаемое значение:

нет

Включаемые файлы:

```
#include "wdmcan.h"
```

Примечание:

Бит №4 в режиме BasicCAN не маскирует соответствующую причину прерывания (выход контроллера из режима "сна"). Эта причина в режиме BasicCAN немаскируется и всегда приводит к возникновению прерывания от платы.

GetStatus

Описание:

Получить состояние CAN-контроллера.

Прототип:

unsigned short GetStatus(void)

Входные параметры:

нет

Возвращаемое значение:

Бит	Описание
7	Состояние шины лог.1 – канал отключен от CAN – шины лог.0 – канал участвует в работе CAN – шины
6	Наличие ошибок лог.1 – по крайней мере один из счётчиков ошибок достиг предельного значения лог.0 – ни один из счётчиков ошибок не достиг предельного значения
5	Состояние передачи лог.1 – CAN-контроллер передает данные лог.0 – передачи данных нет
4	Состояние приёма лог.1 – CAN-контроллер принимает данные лог.0 – приема данных нет
3	Завершенность передачи лог.1 – последняя передача успешно завершена лог.0 – последняя передача была не завершена
2	Доступность буфера передачи лог.1 – буфер передачи доступен лог.0 – буфер передачи не доступен
1	Переполнение приемного буфера лог.1 – произошло переполнение приёмного буфера лог.0 – переполнения приёмного буфера нет
0	Состояние буфера приема лог.1 – буфер приема содержит принятое сообщение лог.0 – буфер приема пуст

Включаемые файлы:

#include "wdmcan.h"

Примечание:

нет

SetTxBuffer

Описание:

Запись кадра в буфер передачи.

Прототип:

unsigned short SetTxBuffer (pRX_TX_Buffer)

Передаваемый параметр:

указатель на структуру RX_TX_Buffer

Возвращаемое значение:

- | | | |
|---|---|--|
| 0 | - | функция успешно завершена |
| 1 | - | буфер передачи не доступен |
| 2 | - | одно или несколько полей структуры задано некорректно |
| 3 | - | была попытка передать расширенный кадр (с 29-битным идентификатором), когда канал находится в режиме BasicCAN. |

Включаемые файлы:

#include "wdmcan.h"

Примечание:

Описание структуры RX_TX_Buffer приведено после описания функций.

GetRxBuffer

Описание:

Получить содержимое буфера приёма

Прототип:

unsigned short GetRxBuffer(pRX_TX_Buffer)

Передаваемый параметр:

Указатель на структуру RX_TX_Buffer

Возвращаемое значение:

Всегда возвращается 0

Включаемые файлы:

#include "wdmcan.h"

Примечание:

1. Функция возвратит содержимое буфера приема, даже если этот буфер пустой (содержит случайные данные). Для получения принятого кадра перед вызовом этой функции необходимо использовать функцию GetStatus.
2. Описание структуры RX_TX_Buffer приведено после описания функций.

SetCommand

Описание:

Передача CAN-контроллеру команды управления.

Прототип:

void SetCommand(unsigned short Command)

Передаваемый параметр:

Бит	Описание
7	не используется
6	не используется
5	не используется
4	Переход в режим "сна" лог.1 – контроллер переходит в режим "сна"
3	Сброс признака переполнения буфера приема лог.1 – признак "Переполнение буфера приёма" будет сброшен.
2	Освобождение буфера приема лог.1 – буфер приема освобождается
1	Прервать передачу лог.1 – если был запрос на передачу кадра из буфера передачи и передача еще не началась, этот запрос будет снят.
0	Запрос на передачу кадра из буфера передачи лог.1 – кадр из буфера передачи будет передан по шине

Возвращаемое значение:

нет

Включаемые файлы:

#include "wdmcan.h"

Примечание:

нет

GetEventData

Описание:

Возвращает причину прерывания (из буфера драйвера).

Прототип:

unsigned short GetEventData(pTEEventData Buffer)

Передаваемый параметр:

В структуру TEventData драйвер помещает причину прерывания и, в случае приема данных, принятый пакет.

Возвращаемое значение:

Всегда 0

Включаемые файлы:

#include "wdmcan.h"

Примечание:

нет

Описание структуры RX_TX_Buffer

Описание:

Структура предназначена для чтения/записи CAN-кадров.

Прототип:

```
typedef struct
{
    unsigned short FF;
    unsigned short sID;
    unsigned long extID;
    unsigned short RTR;
    unsigned short DLC;
    unsigned short DataByte[8];
}RX_TX_Buffer, *pRX_TX_Buffer;
```

Описание полей:

Поле структуры	Допустимые значения	Описание
FF	[0:1]	0 – стандартный кадр 1 – расширенный кадр
sID	[0:7FFh]	стандартный 11-битный идентификатор (если поле FF=1 может иметь любое значение)
extID	[0:1FFFFFFFh]	стандартный 29-битный идентификатор (если поле FF=0 может иметь любое значение)
RTR	[0:1]	RTR-бит кадра
DLC	[0:8]	количество байт данных в кадре
DataByte[8]	[00h:FFh]	массив байт данных

Примечание:

Поле DLC определяет реальное количество байт данных массива DataByte[], которое будет передано/принято в/из буфер передачи/буфера приёма.

Функции для режима BasicCAN

B_SetInputFilter

Описание:

Настройка входного фильтра при приёме кадров по шине CAN для режима BasicCAN.

Прототип:

unsigned short B_SetInputFilter(unsigned short mask, unsigned short kod)

Входные параметры:

mask - маска

kod - идентификатор входного фильтра

Возвращаемое значение:

0 - функция завершена успешно

1 - некорректный (не BasicCAN) режим работы контроллера канала

Включаемые файлы:

#include "wdmcan.h"

Примечание:

Таблица, поясняющая функционирование входного фильтра.

	Бит										
	10	9	8	7	6	5	4	3	2	1	0
Идентификатор кадра	1	0	1	0	1	0	1	0	1	0	1
	Бит										
	7	6	5	4	3	2	1	0			
Маска	1	1	1	1	0	0	0	0			
Идентификатор входного фильтра	x	x	x	x	1	0	1	0			
Результат сравнения	1	1	1	1	1	1	1	1			

x – любое значение.

Входной фильтр функционирует следующим образом:

Биты 3-10 идентификатора кадра сравниваются на идентичность (поразрядно) с идентификатором входного фильтра. Если в соответствующем разряде маски стоит лог.1, то биты идентификатора кадра и идентификатора входного фильтра в этом разряде не сравниваются и результат сравнения в данном разряде принимается равным лог.1. Кадр проходит приемный фильтр, если результат сравнения во всех разрядах содержит лог.1.

Пример:

1. В таблице выше показан пример сравнения идентификатора кадра в приёмном фильтре. Результат сравнения положительный и кадр будет записан в буфер приёма.

2. Для приема всех кадров необходимо установить маску равную FFh (значение идентификатора входного фильтра можно установить любое).

Функции для режима PeliCAN

P_SetRxErrorCounter

Описание:

Установка счетчика ошибок при приеме по шине CAN.

Прототип:

unsigned short P_SetRxErrorCounter(unsigned short)

Входные параметры:

Новое значение счётчика ошибок при приёме

Возвращаемое значение:

0	-	функция успешно завершена
1	-	некорректный (не PeliCAN) режим работы контроллера канала

Включаемые файлы:

#include "wdmcan.h"

Примечание:

нет

P_GetRxErrorCounter

Описание:

Получение счетчика ошибок при приёме по шине CAN.

Прототип:

int P_GetRxErrorCounter(void)

Входные параметры:

нет

Возвращаемое значение:

>=0	-	значение счётчика ошибок при приёме по шине CAN
-1	-	некорректный (не PeliCAN) режим работы контроллера канала

Включаемые файлы:

#include "wdmcan.h"

Примечание:

нет

P_SetTxErrorCounter

Описание:

Установка счетчика ошибок при передаче по шине CAN.

Прототип:

unsigned short P_SetTxErrorCounter(unsigned short)

Входные параметры:

новое значение счётчика ошибок при передаче

Возвращаемое значение:

0	-	функция успешно завершена
1	-	некорректный (не PeliCAN) режим работы контроллера канала

Включаемые файлы:

#include "wdmcan.h"

Примечание:

нет

P_GetTxErrorCounter

Описание:

Получение счетчика ошибок при передаче по шине CAN.

Прототип:

int P_GetTxErrorCounter(void)

Входные параметры:

нет

Возвращаемое значение:

>=0	-	значение счётчика ошибок при передаче по шине CAN
-1	-	некорректный (не PeliCAN) режим работы контроллера канала

Включаемые файлы:

#include "wdmcan.h"

Примечание:

нет

P_SetErrorWarningLimit

Описание:

Установка верхнего предела счетчиков ошибок.

Прототип:

unsigned short P_SetErrorWarningLimit(unsigned short)

Входные параметры:

новый верхний предел счётчиков ошибок

Возвращаемое значение:

0	-	функция успешно завершена
1	-	некорректный (не PeliCAN) режим работы контроллера канала

Включаемые файлы:

#include "wdmcan.h"

Примечание:

1. Если хоть один из счетчиков ошибок (при приеме или передаче) превысит значение предела заданного этой функцией, произойдет установка бита "Наличие ошибок" (см. функцию GetStatus).
2. По умолчанию верхний предел счётчиков ошибок равен 96.

P_GetErrorWarningLimit

Описание:

Получение верхнего предела счетчиков ошибок.

Прототип:

int P_GetErrorWarningLimit(void)

Входные параметры:

нет

Возвращаемое значение:

>=0	-	значение верхнего предела счётчиков ошибок
-1	-	некорректный (не PeliCAN) режим работы контроллера канала

Включаемые файлы:

#include "wdmcan.h"

Примечание:

нет

P_GetArbitrationLostCapture

Описание:

Получение номера бита, на котором был проигран арбитраж.

Прототип:

```
int P_GetArbitrationLostCapture(void)
```

Входные параметры:

нет

Возвращаемое значение:

0	-	Бит №1 идентификатора
.....		
10	-	Бит №11 идентификатора
11	-	бит RTR стандартного идентификатора
12	-	бит IDE
13	-	Бит №12 расширенного идентификатора
.....		
30	-	Бит №29 расширенного идентификатора
31	-	бит RTR расширенного кадра
-1	-	некорректный (не PeliCAN) режим работы контроллера канала

Включаемые файлы:

```
#include "wdmcan.h"
```

Примечание:

нет

P_GetRxMessageCounter

Описание:

Получение количества принятых (и не прочитанных) кадров в буфере приёма.

Прототип:

```
int P_GetRxMessageCounter(void)
```

Входные параметры:

нет

Возвращаемое значение:

≥ 0	-	количество принятых (и не прочитанных) кадров в буфере приёма.
-1	-	некорректный (не PeliCAN) режим работы контроллера канала

Включаемые файлы:

```
#include "wdmcan.h"
```

Примечание:

Счетчик автоматически увеличивается после приёма кадра из шины и автоматически уменьшается после выполнения команды "Освободить буфер приёма" (см. функцию SetCommand).

P_GetErrorCode

Описание:

Получение типа и местоположения ошибки при обмене с шиной CAN.

Прототип:

```
int P_GetErrorCode(void)
```

Входные параметры:

нет

Возвращаемое значение:

>=0 - тип и местоположение ошибки (см. примечание).
-1 - некорректный (не PeliCAN) режим работы контроллера канала

Включаемые файлы:

```
#include "wdmcan.h"
```

Примечание:

Назначение бит возвращаемого значения:

Бит	7	6	5	4	3	2	1	0
	(см. таблицу ниже)		Dir	(см. таблицу ниже)				

Бит Dir определяет направление обмена, при котором произошла ошибка.

лог. 0 – ошибка произошла в течение передачи

лог. 1 – ошибка произошла в течение приема

Интерпретация бит №0-№4 возвращаемого значения:

Бит					Местоположение ошибки
4	3	2	1	0	
0	0	0	1	1	"начало кадра"
0	0	0	1	0	Идентификатор: биты 28-21
0	0	1	1	0	Идентификатор: биты 20-18
0	0	1	0	0	бит SRTR
0	0	1	0	1	бит IDE
0	0	1	1	1	Идентификатор: биты 17-13
0	1	1	1	1	Идентификатор: биты 12-5
0	1	1	1	0	Идентификатор: биты 4-0
0	1	1	0	0	бит RTR
0	1	1	0	1	зарезервированный бит 1
0	1	0	0	1	зарезервированный бит 0
0	1	0	1	1	"код длины данных"
0	1	0	1	0	поле данных
0	1	0	0	0	CRC последовательность
1	1	0	0	0	разделитель CRC
1	1	0	0	1	"интервал подтверждения"
1	1	0	1	1	"разделитель подтверждения"
1	1	0	1	0	"конец кадра"
1	0	0	1	0	"перерыв" на шине
1	0	0	0	1	флаг активной ошибки
1	0	1	1	0	флаг пассивной ошибки
1	0	0	1	1	tolerate dominant bits
1	0	1	1	1	разделитель ошибки
1	1	1	0	0	флаг переполнения

Интерпретация бит №6,7 возвращаемого значения:

Бит		Тип ошибки
7	6	
0	0	битовая ошибка
0	1	ошибка формы
1	0	ошибка заполнения
1	1	другой тип ошибки

P_SetInputFilter

Описание:

Настройка входного фильтра при приёме кадров по шине CAN для режима PeliCAN.

Прототип:

```
unsigned short P_SetInputFilter(unsigned short mode, unsigned short ACR0,  
                                unsigned short ACR1, unsigned short ACR2, unsigned short  
                                ACR3, unsigned short AMR0, unsigned short AMR1, unsigned  
                                short AMR2, unsigned short AMR3)
```

Входные параметры:

mode - режим функционирования входного фильтра:
1 – single filter
0 – dual filter
ACR0 –ACR3 – регистры идентификатора входного фильтра
AMR0-AMR3 – регистры маски входного фильтра

Возвращаемое значение:

0 - функция завершена успешно
1 - некорректный (не PeliCAN) режим работы контроллера канала
2 - параметр mode задан не корректно.

Включаемые файлы:

```
#include "wdmcan.h"
```

Примечание:

Пример:

1. Настройка входного фильтра в режиме PeliCAN описана в описании контроллера SJA1000 (описание контроллера входит в комплект поставки к плате). Параметры ACRx и AMRx представляют собой значения, записываемые в одноименные регистры контроллера.
2. Для приема всех кадров необходимо установить маску равную FFh (значение идентификатора входного фильтра можно установить любое).

Описание структуры TEventData

Описание:

Структура, описывающая прерывание, возникшее от платы CAN-200

Прототип:

```
typedef struct
{
    RX_TX_Buffer    rxtxbuf;
    UCHAR           IntrID;
} TEventData, *pTEventData;
```

Описание полей:

IntrID:

Содержимое регистра идентификации прерывания CAN-контроллера. Если поле IntrID=1 (возникло прерывание по приему кадра), то структура rxtxbuf содержит принятый кадр.

rxtxbuf:

Структура, содержит принятый кадр данных. Поля структуры содержат действительные значения только при поле IntrID равным 1.

Включаемые файлы:

```
#include "wdmcan.h"
```

Примечание:

нет

Пример обработки прерываний

```
#include <stdio.h>
#include <conio.h>
#include <windows.h>
#include "wdmcan.h"

SECURITY_ATTRIBUTES sa4Event;
HANDLE hEvent;
TEventData evd;

void main(void)
{
    sa4Event.nLength = sizeof(sa4Event);
    sa4Event.lpSecurityDescriptor = NULL;
    sa4Event.bInheritHandle = TRUE;

    if ( !(hEvent = CreateEvent(&sa4Event, TRUE, FALSE, NULL)) )
    {
        printf ("CreateEvent failed\n");
        return;
    }
    DefEvent(hEvent,TRUE);

    //.....

    switch (WaitForSingleObject( hEvent, 200 ))
    {
        case WAIT_OBJECT_0:
            ResetEvent(hEvent);
            printf("We got event \n");
            GetEventData(&evd);
            break;

        case WAIT_TIMEOUT:
        default:
            printf("We didn't get event\n");
            break;
    }
}
```

В драйвере предусмотрена буферизация прерываний. Сведения о сформировавшихся прерываниях, которые не были обработаны приложением, сохраняются в драйвере. Буферизация обеспечивает сохранение информации о 256 прерываниях. При переполнении буфера формирование драйвером сообщений приложению о прерывании приостанавливается. Последующий вызов функции GetEventData вернет информацию о последнем прерывании.