

Описание библиотеки
функций
для плат серии CAN-200

Руководство пользователя.
v1.1

Операционная система: DOS 6.22 и выше

Оглавление

Назначение	3
Условия работы	3
Введение	4
Функции инициализации и конфигурирования платы	5
chBaseAddress	5
HardReset.....	5
VME_Enable.....	6
VME_Disable.....	6
GetVMEStatusID	7
SetVMEStatusID.....	7
Функции для режимов BasicCAN и PeliCAN	8
SetWorkMode	8
GetWorkMode.....	8
SetDriverMode.....	9
SetCANSpeed	9
SetInterruptSource.....	10
GetInterruptSource	11
GetStatus	12
SetTxBuffer.....	13
GetRxBuffer.....	13
SetCommand	14
Описание структуры Buffer.....	15
Функции для режима BasicCAN	16
B_SetInputFilter.....	16
Функции для режима PeliCAN	17
P_SetRxErrorCounter	17
P_SetTxErrorCounter.....	18
P_GetTxErrorCounter	18
P_SetErrorWarningLimit	19
P_GetErrorWarningLimit	19
P_GetArbitrationLostCapture	20
P_GetRxMessageCounter	21
P_GetErrorCode.....	22
P_SetInputFilter	24

Назначение

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

Условия работы

Установленное оборудование:

IBM PC совместимый ПК 80386 и выше.

Для платы CAN-200-6U: для корректной работы библиотеки функций требуется мост PCI-VME фирмы Tundra Semiconductor серии Universe.

Программное обеспечение:

Компилятор Borland C++ версии 3.1

(для работы с платой CAN-200-6U необходимо работать в режиме DPMI16)

Поддерживаемые операционные системы:

- MSDOS версии 6.22 и выше

Поддерживаемые платы:

- CAN-200PC
- CAN-200MP
- CAN-200PC104
- CAN-200PCI
- CAN-200-6U
- CAN-200-104PCI

Примечание:

1. При построении исполняемого модуля для работы с платой CAN-200-6U необходимо использовать библиотеку `can200v.lib` и заголовочный файл `can200v.h`. Для остальных плат необходимо использовать библиотеку `can200.lib` и заголовочный файл `can200.h`.
2. 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, CAN-200-104PCI) 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 (29-битным) идентификатором.

Функции инициализации и конфигурирования платы

chBaseAddress

Описание:

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

Прототип:

```
void chBaseAddress(unsigned int)
```

Входные параметры:

Базовый адрес CAN-канала

Возвращаемое значение:

нет

Включаемые файлы:

Для всех плат, кроме CAN-200-6U: `#include "can200.h"`

Для платы CAN-200-6U: `#include "can200v.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 "can200.h"`

Примечание:

Функция не работает с платами CAN-200PCI, CAN-200-104PCI и CAN-200-6U.

VME_Enable

Описание:

Функция инициализации моста PCI-VME.

Функция используется только при работе с платами CAN-200-6U.

Прототип:

unsigned int VME_Enable(void)

Передаваемые параметры:

нет

Возвращаемое значение:

VME_OK – функция выполнена без ошибок

VME_ERR_INITIALIZATION_ERROR – ошибка инициализации моста PCI-VME

Включаемые файлы:

#include "can200v.h"

Примечание:

Эта функция должна однократно вызываться перед вызовами других функций драйверной библиотеки.

VME_Disable

Описание:

Функция запрета доступа к мосту PCI-VME.

Функция используется только при работе с платами CAN-200-6U.

Прототип:

void VME_Disable(void)

Передаваемые параметры:

нет

Возвращаемое значение:

нет

Включаемые файлы:

#include "can200v.h"

Примечание:

Эта функция должна однократно вызываться после завершения вызовов всех других функций драйверной библиотеки.

GetVMEStatusID

Описание:

Получить вектор прерывания шины VME для выбранного канала.
Функция используется только при работе с платами CAN-200-6U.

Прототип:

unsigned char GetVMEStatusID(void)

Передаваемые параметры:

нет

Возвращаемое значение:

Значение вектора прерывания шины VME выбранного канала.

Включаемые файлы:

Для всех плат, кроме CAN-200-6U: #include "can200.h"
Для платы CAN-200-6U: #include "can200v.h"

SetVMEStatusID

Описание:

Установить вектор прерывания шины VME для выбранного канала.
Функция используется только при работе с платами CAN-200-6U.

Прототип:

void SetVMEStatusID(unsigned char)

Передаваемые параметры:

Значение вектора прерывания шины VME.

Возвращаемое значение:

нет

Включаемые файлы:

Для всех плат, кроме CAN-200-6U: #include "can200.h"
Для платы CAN-200-6U: #include "can200v.h"

Функции для режимов BasicCAN и PeliCAN

SetWorkMode

Описание:

Изменить режим работы CAN-контроллера.

Прототип:

unsigned char SetWorkMode(unsigned char)

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:

Для всех плат, кроме CAN-200-6U: #include "can200.h"

Для платы CAN-200-6U: #include "can200v.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 char GetWorkMode(void)

Входные параметры:

нет

Возвращаемое значение:

BasicCAN - контроллер находится в режиме BasicCAN

PeliCAN - контроллер находится в режиме PeliCAN

Включаемые файлы:

Для всех плат, кроме CAN-200-6U: #include "can200.h"

Для платы CAN-200-6U: #include "can200v.h"

Примечание:

CAN-контроллер PCA82C200 всегда находится в режиме BasicCAN.

SetDriverMode

Описание:

Функция настройки выходного формирователя CAN-контроллера.

Прототип:

void SetDriverMode (unsigned char)

Входные параметры:

Параметр должен принимать значение 1Bh

Возвращаемое значение:

нет

Включаемые файлы:

Для всех плат, кроме CAN-200-6U: #include "can200.h"

Для платы CAN-200-6U: #include "can200v.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 char 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:

Для всех плат, кроме CAN-200-6U: #include "can200.h"

Для платы CAN-200-6U: #include "can200v.h"

Примечание:

нет

SetInterruptSource

Описание:

Разрешение источников прерывания.

Прототип:

```
void SetInterruptSource(unsigned char)
```

Входные параметры:

Байт маски прерывания.

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

Возвращаемое значение:

нет

Включаемые файлы:

Для всех плат, кроме CAN-200-6U: `#include "can200.h"`

Для платы CAN-200-6U: `#include "can200v.h"`

Примечание:

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

GetInterruptSource

Описание:

Идентификация источника прерывания.

Прототип:

unsigned char GetInterruptSource(void)

Входные параметры:

нет

Возвращаемое значение:

Байт, идентифицирующий источник прерывания:

Бит	Описание
7	Прерывание при ошибках на шине (только для режима PeliCAN) лог.1 – контроллер находит ошибки при обмене по CAN-шине
6	Прерывание при проигрыше арбитража (только для режима PeliCAN) лог.1 – контроллер проиграл арбитраж на шине
5	Переход в "error passive status" (только для режима PeliCAN) лог.1 – контроллер перешел в состояние "error passive status"
4	Прерывание при выходе из состояния "сна" (только для режима PeliCAN) лог.1 – канал находился в состоянии "сна" и была активность на шине
3	Прерывание при переполнении буфера приёма лог.1 – бит "Переполнение приемного буфера" перешел из лог. 0 в состояние лог.1 (см. функцию GetStatus)
2	Прерывание по ошибкам на шине лог.1 – было изменение бит "Наличие ошибок" или "Состояние шины" (см. функцию GetStatus)
1	Прерывание после передачи кадра лог.1 – был переход буфера передачи из состояния "не доступен" в состояние "доступен"
0	Прерывание по приёму кадра лог.1 – буфер приема не пуст

Включаемые файлы:

Для всех плат, кроме CAN-200-6U: #include "can200.h"

Для платы CAN-200-6U: #include "can200v.h"

Примечание:

см. также функцию SetInterruptSource

GetStatus

Описание:

Получить состояние CAN-контроллера.

Прототип:

unsigned char 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:

Для всех плат, кроме CAN-200-6U: #include "can200.h"

Для платы CAN-200-6U: #include "can200v.h"

Примечание:

нет

SetTxBuffer

Описание:

Запись кадра в буфер передачи.

Прототип:

unsigned char SetTxBuffer (struct Buffer*)

Передаваемый параметр:

указатель на структуру 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:

Для всех плат, кроме CAN-200-6U: #include "can200.h"

Для платы CAN-200-6U: #include "can200v.h"

Примечание:

Описание структуры Buffer приведено после описания функций.

GetRxBuffer

Описание:

Получить содержимое буфера приёма

Прототип:

struct Buffer *GetRxBuffer(void)

Передаваемый параметр:

нет

Возвращаемое значение:

Указатель на структуру Buffer.

Включаемые файлы:

Для всех плат, кроме CAN-200-6U: #include "can200.h"

Для платы CAN-200-6U: #include "can200v.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 Buffer приведено после описания функций.

.

SetCommand

Описание:

Передача CAN-контроллеру команды управления.

Прототип:

void SetCommand(unsigned char)

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:

Для всех плат, кроме CAN-200-6U: #include "can200.h"

Для платы CAN-200-6U: #include "can200v.h"

Примечание:

нет

Описание структуры Buffer

Описание:

Структура предназначена для чтения/записи CAN-кадров.

Прототип:

```
struct Buffer
{
    unsigned char FF;
    unsigned int sID;
    unsigned long extID;
    unsigned char RTR;
    unsigned char DLC;
    unsigned char DataByte[8];
};
```

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 char B_SetInputFilter(unsigned char mask, unsigned char 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:

Для всех плат, кроме CAN-200-6U: #include "can200.h"

Для платы CAN-200-6U: #include "can200v.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 char P_SetRxErrorCounter(unsigned char)
```

Входные параметры:

Новое значение счётчика ошибок при приёме

Возвращаемое значение:

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

Включаемые файлы:

Для всех плат, кроме CAN-200-6U:	#include "can200.h"
Для платы CAN-200-6U:	#include "can200v.h"

Примечание:

нет

P_GetRxErrorCounter

Описание:

Получение счетчика ошибок при приеме 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:

Для всех плат, кроме CAN-200-6U:	#include "can200.h"
Для платы CAN-200-6U:	#include "can200v.h"

Примечание:

нет

P_SetTxErrorCounter

Описание:

Установка счетчика ошибок при передаче по шине CAN.

Прототип:

unsigned char P_SetTxErrorCounter(unsigned char)

Входные параметры:

Новое значение счётчика ошибок при передаче

Возвращаемое значение:

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

Включаемые файлы:

Для всех плат, кроме CAN-200-6U: #include "can200.h"
Для платы CAN-200-6U: #include "can200v.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:

Для всех плат, кроме CAN-200-6U: #include "can200.h"
Для платы CAN-200-6U: #include "can200v.h"

Примечание:

нет

P_SetErrorWarningLimit

Описание:

Установка верхнего предела счетчиков ошибок.

Прототип:

unsigned char P_SetErrorWarningLimit(unsigned char)

Входные параметры:

Новый верхний предел счётчиков ошибок

Возвращаемое значение:

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

Включаемые файлы:

Для всех плат, кроме CAN-200-6U:	#include "can200.h"
Для платы CAN-200-6U:	#include "can200v.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:

Для всех плат, кроме CAN-200-6U:	#include "can200.h"
Для платы CAN-200-6U:	#include "can200v.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:

Для всех плат, кроме CAN-200-6U:	#include "can200.h"
Для платы CAN-200-6U:	#include "can200v.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:

Для всех плат, кроме CAN-200-6U:	<code>#include "can200.h"</code>
Для платы CAN-200-6U:	<code>#include "can200v.h"</code>

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:

Для всех плат, кроме CAN-200-6U: #include "can200.h"
Для платы CAN-200-6U: #include "can200v.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 char P_SetInputFilter(unsigned char mode, unsigned char ACR0,  
                               unsigned char ACR1, unsigned char ACR2, unsigned char  
                               ACR3, unsigned char AMR0, unsigned char AMR1, unsigned  
                               char AMR2, unsigned char 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:

Для всех плат, кроме CAN-200-6U: #include "can200.h"
Для платы CAN-200-6U: #include "can200v.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).