# Библиотека

# SLMCP

# Руководство программиста

## *Версия документа 1.5*

ООО «НПП Славна»
624250 Свердловская обл. г.Заречный, ул.Мира, 35
тел. (34377) 736-91
тел/факс (34377) 341-51
E-mail npp@slavna.ru
http://www.slavna.ru

# *Содержание*

## *Введение*

Библиотека предназначена для доступа к сети CAN на канальном уровне и разработана для применения во встраиваемых приложениях под управлением операционной системы Windows XP и Windows 2000..

## *Возможности библиотеки.*

Библиотека поддерживает:

- работу с несколькими одновременно работающими конвертерами;
- стандартный и расширенный протокол CAN (11-битный и 29-битный идентификаторы);
- настройку аппаратного фильтра контроллера;
- настройку скорости передачи;
- поддержку очереди сообщений на передачу и FIFO-буфер на приём;
- настройку временных характеристик конвертера;
- поддержку доступа к аппаратным средствам CAN контроллера.

## *Описание API.*

Каждый CAN конвертер идентифицируется целым числом – номером канала – начиная с «1». *Номер «0», используемый в предыдущей версии драйвера, больше не поддерживается.* Каждый конвертер рассматривается библиотекой как отдельное устройство. С каждым конвертером ассоциирована своя очередь на передачу и свой FIFO буфер на приём.

Программный интерфейс библиотеки состоит из трёх групп функций:

- Функции загрузки и выгрузки библиотеки.
- Базовые функции, предназначенные для конфигурирования конвертера и для приёма – отправки CAN пакетов.
- Дополнительные функции, предназначенные для расширенной работы с CAN контроллером конвертера.

## *Список функций библиотеки.*

### Загрузка и выгрузка библиотеки.

```
bool  CnOpenLibrary();
void  CnCloseLibrary();
```

### Базовые функции:

```
ULONG CnOpen( UCHAR aNumDev);
ULONG CnClose( ULONG Handle);
ULONG CnInit( ULONG Handle, PUCHAR pDevInfo, bool aBlockRead);
ULONG CnSetMaskFilters( ULONG Handle, PULONG pMF);
ULONG CnSetBaudRate( ULONG Handle, UCHAR aBaud);
ULONG CnRecievePackets( ULONG Handle, PCANDATA pBuf, PULONG
pRecievePackets);
```

```
    ULONG CnSendPackets( ULONG Handle, PCANDATA pBuf, ULONG aSendPackets, PUCHAR
pStatus);
    ULONG CnRecieveEnable( ULONG Handle);
    ULONG CnRecieveDisable( ULONG Handle);
    ULONG CnGetSerialNumber( ULONG Handle, PUCHAR pSn, UCHAR aLenBuf);
    ULONG CnGetDevParameters( ULONG Handle, PDEVPARAM pBuf);
    ULONG CnSetDevParameters ( ULONG Handle, PDEVPARAM pBuf);
```

### Дополнительные фунции:

```
ULONG CnSetRegisters( ULONG Handle, UCHAR aAddr, UCHAR aBytes, PUCHAR pBuf);
ULONG CnGetRegisters( ULONG Handle, UCHAR aAddr, UCHAR aBytes, PUCHAR pBuf);
```

### *Структура CAN пакета.*

Библиотека работает со структурой CAN пакета, имеющей следующий прототип:

```
typedef
struct
{ ULONG ID;       /* идентификатор пакета. */
  UCHAR DLC;      /* количество байт данных в пакете*/
  UCHAR d[8];     /* массив данных*/
  UCHAR reserv1;  /* зарезервировано */
  UCHAR reserv2;  /* зарезервировано*/
  UCHAR status;   /* поле статуса*/
} TCANDATA, *PCANDATA;
```

Поле **ID** - идентификатор пакета. Состоит из 4х байт и имеет следующую структуру:
0-й байт - младший

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 |

старший ← разряд → младший

1-й байт

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 |

2-й байт

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| EID20<br>SID2 | EID19<br>SID1 | EID18<br>SID0 | SRTR | EXT | - | EID17 | EID16 |

3-й байт - старший

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| EID28<br>SID10 | EID274<br>SID9 | EID26<br>SID8 | EID25<br>SID7 | EID24<br>SID6 | EID23<br>SID5 | EID22<br>SID4 | EID21<br>SID3 |

где

**SIDn** –**n**-й бит стандартного идентификатора,

**EIDn** – **n**-й бит расширенного идентификатора.

**EXT** – бит признака расширенного идентификатора.

**SRTR** – у принятого пакета бит признака удалённого запроса (Remote Transmit Request) при использовании стандартного идентификатора пакета. Имеет правильное значение когда бит **EXT** равен 0.

Для работы с полем идентификатора определены следующие константы:

```
    #define flag_RX_RTR_St  0x100000 /* признак запроса удалённой передачи
(Remote Transmit Request) принятого пакета со стандартным идентификатором */
    #define flag_IDEXT      0x80000  /* признак расширенного идентификатора
пакета */
```

Поле **DLC –** «Data Length Code» имеет размер 1 байт. В этом поле определены следующие биты:

| - | RTR | - | - | DLC3 | DLC2 | DLC1 | DLC0 |
|---|---|---|---|---|---|---|---|

старший ← разряд → младший

Где

**DLC0 -:- DLC3** – число байт данных в пакете. Число не должно быть больше 8.

**RTR** – на передачу **–** передаваемый пакет является удалённым запросом, на приём – признак того, что пакет является удалённым запросом если идентификатор пакета имеет расширенный вариант (бит **RTR** определён только тогда, когда бит **EXT** в идентификаторе пакета равен 1).

Для работы с полем **RTR** определены следующие константы

```
#define flag_RTR        0x40 /* признак запроса удалённой передачи (Remote Transmit Request) */
#define flag_RX_RTR_Ext 0x40 /* признак запроса удалённой передачи (Remote Transmit Request) принятого пакета с расширенным идентификатором */
```

Поля **reserv1** и **reserv1** не используются.

Поле **status** статуса имеет размер 1 байт и. В этом поле определены следующие биты:

| - | - | - | - | | - | SO | HO |
|---|---|---|---|---|---|---|---|

старший ← разряд → младший

Где

**SO** - «software overflow» - признак переполнения FIFO буфера в драйвере. Не вошедшие в буфер пакеты потеряны.

**HO** - «hardware overflow» - признак переполнения буфера в контроллере. Имеются потерянные пакеты.

Определены следующие биты поля статуса:

```
stat_HD_OVERFLOW  = 0x01 - признак аппаратного переполнения ( в контроллере).
stat_SOFT_OVERFLOW= 0x02 - признак программного переполнения ( буфера в драйвере).
```

## *Детальное описание функций.*

## *Функции загрузки и выгрузки библиотеки.*

### bool CnOpenLibrary();

Функция открывает библиотеку.

*Возвращаемое значение*:

TRUE – успешное выполнение.

FALSE – ошибка. Не найден файл **slmcp.dll**.

### void CnCloseLibrary();

Функция закрывает библиотеку.

## *Базовые функции.*

### ULONG Handle = CnOpen( UCHAR aNumDev);

Функция открывает канал ввода-вывода. Она вызывается один раз для каждого конвертера в самом начале работы с каналом (конвертером).

*Параметры*:
**aNumDev** – номер канала ввода-вывода. Возможные значения 1-:-255.
*Возвращаемое значение*:
**Handle** – ссылка на открытый канал. Этот параметр необходим для работы остальных функций.
0 – ошибка

Примечание. Значение номера канала «0» больше не поддерживается. Назначение номера канала для конвертера описано в документации на конвертер в разделе «установка программного обеспечения – выбор номера канала».

### ULONG CnClose( ULONG Handle);

Функция закрывает канал ввода-вывода.
*Параметры*:
**Handle** – ссылка на открытый канал
*Возвращаемое значение*:
не 0 – успешное выполнение
0 – ошибка

### ULONG CnInit(ULONG Handle, PUCHAR pDevInfo, bool aBlockRead);

Функция очищает буфера в конвертере и драйвере, инициализирует внутренние данные библиотеки и USB-CAN конвертера и настраивает конвертер в состояние «по умолчанию». Устанавливается скорость канала 1000 KBaud, значение фильтра и маски сбрасываются в «0», разрешается использование стандартных и расширенных идентификаторов пакетов и запрещается приём собственно CAN пакетов.

*Параметры:*
**Handle** – ссылка на открытый канал.
**pDevInfo** – указатель на 1-байтовую переменную, где сохраняется идентификатор установленного в конвертере контроллера. **0x10** – контроллер **MCP-2515**.
**aBlockRead** – «0» - «неблокирующий» / «не 0» - «блокирующий» режим считывания пакетов.
*Возвращаемое значение:*
не 0 – успешное выполнение
0 – ошибка

Примечание. Блокирующий режим означает, что если запрос на чтение данных сделан при пустом буфере, то возврат из запроса производится только при поступлении в буфер хотя бы одного CAN пакета. В неблокирующем режиме возврат из запроса чтения данных выполняется немедленно в не зависимости от количества данных в буфере.

### ULONG CnSetFilterMask(ULONG Handle, PULONG pMF);

Функция устанавливает значения аппаратных маски и фильтров для идентификаторов принимаемых CAN пакетов. Контроллер конвертера будет принимать только те пакеты, у которых биты идентификатора будут совпадать с битами фильтра aFilter, отмеченными установленными в «1» битами aMask.

*Параметры:*
**Handle** – ссылка на открытый канал.
**pMF** – указатель на буфер маски и фильтров.
*Возвращаемое значение:*
не 0 – успешное выполнение
0 – ошибка

Формат буфера маски и фильтров следующий:

```
TMASKORFILTER mf[4];
```

где
mf[0] – информация о канале и режиме приёма,
mf[1] – значение маски,
mf[2] – значение фильтра 0,
mf[3] – значение фильтра 1.

Элементы буфера имеют следующую структуру

```
typedef
struct
{ UCHAR NumChannel;
  UCHAR IDRXMode;
} TCHANNELINFO, *PCHANNELINFO;

typedef
union
{ TCHANNELINFO CnlInfo;
  ULONG maskfilter;
} TMASKORFILTER, *PMASKORFILTER;
```

**mf[0].NumChannel = 0** - номер канала приёма –должен быть всегда 0.
**mf[0].IDRXMode** - режим приёма пакетов.
Режим приёма пакетов определяет для каких идентификаторов будут использоваться фильтры. Для установки режима приема определены следующие константы:

```
#define rxm_FILTER_ID_STD    0x20
#define rxm_FILTER_ID_EXT    0x40
```

Поле может принимать следующие значения:
**0** – маска и фильтры отключены. Принимаются любые пакеты.

**rxm_FILTER_ID_STD** – Будут приниматься пакеты со стандартным идентификатором, удовлетворяющим установленным маске и фильтрам.

**rxm_FILTER_ID_EXT** – Будут приниматься пакеты с расширенным идентификатором, удовлетворяющим установленным маске и фильтрам.

**rxm_FILTER_ID_STD | rxm_FILTER_ID_EXT** – Будут приниматься пакеты со стандартным и расширенным идентификатором, удовлетворяющим установленным маске и фильтрам.

**mf[1]** – маска.

Формат маски – 4х байтовое слово, биты которого имеют следующее значение:

младший -0й байт

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 |

старший ← разряд → младший

1й байт

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 |

2й байт

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| EID20<br>SID2 | EID19<br>SID1 | EID18<br>SID0 | - | - | - | EID17 | EID16 |

3й байт - старший

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| EID28<br>SID10 | EID274<br>SID9 | EID26<br>SID8 | EID25<br>SID7 | EID24<br>SID6 | EID23<br>SID5 | EID22<br>SID4 | EID21<br>SID3 |

где

**SIDn** –**n**-й бит стандартного идентификатора,

**EIDn** – **n**-й бит расширенного идентификатора.

**mf[2]** фильтр 0,

**mf[3]** фильтр 1.

Формат фильтров аналогичен формату маски. Добавлен бит **EXT**– признак расширенного идентификатора.

младший -0й байт

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 |

старший ← разряд → младший

1й байт

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 |

2й байт

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| EID20<br>SID2 | EID19<br>SID1 | EID18<br>SID0 | - | EXT | - | EID17 | EID16 |

3й байт - старший

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| EID28<br>SID10 | EID274<br>SID9 | EID26<br>SID8 | EID25<br>SID7 | EID24<br>SID6 | EID23<br>SID5 | EID22<br>SID4 | EID21<br>SID3 |

где

**SIDn** –**n**-й бит стандартного идентификатора;

**EIDn** – **n**-й бит расширенного идентификатора;

**EXT** – признак расширенного идентификатора;

Если установленный режим приёма **IDRXMode** не соответствует установленным фильтрам, то фильтры не используются.

### ULONG CnRecieveEnable( ULONG Handle);

Функция разрешает приём CAN пакетов в конвертере.
*Параметры:*
**Handle** – ссылка на открытый канал.
*Возвращаемое значение:*
не 0 – успешное выполнение
0 – ошибка

### ULONG CnRecieveDisable(ULONG Handle);

Функция разрешает приём CAN пакетов в конвертере.

*Параметры:*
**Handle** – ссылка на открытый канал.
*Возвращаемое значение:*
не 0 – успешное выполнение
0 – ошибка

### ULONG CnSetBaudRate( ULONG Handle, UCHAR aBaud);

Функция устанавливает скорость передачи CAN пакетов по сети.

*Параметры:*
**Handle** – ссылка на открытый канал.
**aBaud** – скорость передачи согласно таблице:

```
0 - cCAN_BD_1000 – 1000K
1 - cCAN_BD_800  - 800K
2 - cCAN_BD_500  - 500K
3 - cCAN_BD_250  - 250K
4 - cCAN_BD_125  - 125K
5 - cCAN_BD_50   - 50K
6 - cCAN_BD_20   - 20K
7 - cCAN_BD_10   - 10K
```

*Возвращаемое значение:*
не 0 – успешное выполнение
0 – ошибка

### ULONG CnRecievePackets( ULONG Handle, PCANDATA pBuf, PULONG pRecievePackets);

Функция приёма пакетов.

*Параметры:*
**Handle** – ссылка на открытый канал.
**pBuf** – указатель на буфер принимаемых пакетов.

**pRecievePackets** – указатель на переменную в которой указан размер буфера пакетов в пакетах (в пакетах  а не в байтах!).

*Возвращаемые значения:*

не 0 – успешное выполнение

0 – ошибка

**pRecievePackets** – по этому указателю размер буфера пакетов будет заменён на количество считанных пакетов (в пакетах).

*Примечание.* Используемый на приём FIFO буфер ассоциирован с конкретным каналом (конвертером) а не приложением. В результате если был разрешён приём пакетов в конвертере и приложение закрылось, то FIFO буфер будет накапливать приходящие пакеты. Приложение, подключаясь вновь к этому же каналу и не вызывая его инициализацию, мажет считать накопленные пакеты.

### ULONG CnSendPackets( ULONG Handle, PCANDATA pBuf, ULONG aSendPackets, PUCHAR pStatus);

Функция отправки пакетов.

*Параметры:*

**Handle** – ссылка на открытый канал.

**pBuf** – указатель на буфер отправляемых пакетов. Максимальное число пакетов в одной транзакции – **0x2000** (8192 пакета)

**aSendPackets** – количество отправляемых пакетов

**pStatus** – указатель на переменную для получения результата отправки.

*Возвращаемые значения:*

не 0 – успешное выполнение

0 – ошибка

Переменная по указателю **pStatus** – принимает значения

0 – успешное выполнение

**stat_TX_TIMEOUT** = **0x10**– таймаут передачи. Причина – обрыв CAN сети или отсутствие в сети другого CAN устройства. Время до возникновения этой ошибки примерно 3 сек на одну транзакцию.

### ULONG CnSerialNumber(ULONG Handle, PUCHAR pSn; UCHAR aLenBuf)

Функция возвращает серийный номер USB-CAN конвертера. Серийный номер возвращается в виде строки ANSI символов, заканчивающихся нулём.

*Параметры:*

**Handle** – ссылка на открытый канал.

**pSn** – указатель на буфер для получения строки.

**aLenBuf** – длина буфера в байтах.

*Возвращаемое значение:*

не 0 – успешное выполнение

0 – ошибка

### ULONG CnSetDevParameters(ULONG Handle, PDEVPARAM pBuf)

Функция изменяет параметры конвертера.

*Параметры:*
**Handle** – ссылка на открытый канал.
**pBuf** – указатель на структуру TDEVPARAM.
*Возвращаемое значение:*
не 0 – успешное выполнение
0 – ошибка

Структура TDEVPARAM имеет следующий прототип:

```
typedef
struct
{ UCHAR size;
  UCHAR microcode;
  UCHAR time_assemble_packets;
  UCHAR timeout_can_transaction;
}TDEVPARAM, *PDEVPARAM;
```

где
**size** – размер этой структуры;
**microcode** – зарезервировано, может быть любое.
**time_assemble_packets** – время пакетирования USB пакетов. Детально описано в документации на конвертер в главе «Программная настройка». Допустимые значения [0-20].
**timeout_can_transaction** - таймаут отправки CAN пакетов. Детально описано в документации на конвертер в главе «Программная настройка». Допустимые значения [1-200].

### ULONG CnGetDevParameters(ULONG Handle, PDEVPARAM pBuf)

Функция считывает параметры конвертера.

*Параметры:*
**Handle** – ссылка на открытый канал.
**pBuf** – указатель на структуру TDEVPARAM.
*Возвращаемое значение:*
не 0 – успешное выполнение
0 – ошибка
Структура TDEVPARAM документирована выше.

**Внимание!** Поле размер буфера (структуры) - **size** - должно быть выставлено **перед вызовом** этой функции. При безошибочном выполнении функции это поле будет содержать требуемый максимальный размер буфера под структуру. Количество принятых байт будет равно размеру выделенного буфера, но не больше максимального размера структуры.

## *Дополнительные функции:*

Дополнительные функции позволяют проводить наиболее гибкую настройку CAN контроллера. Эти функции аппаратно зависимы и реализованы для CAN контроллера

MCP-2515. Назначения регистров контроллера приведено в фирменной документации. Микропрограмма конвертера поддерживает только один буфер на передачу (нулевой) и только один буфер на приём (также нулевой). С помощью дополнительных функций можно осуществить подстройку скорости передачи пакетов, прочитать счётчики ошибок и прочее.

### ULONG CnSetRegisters(ULONG Handle, UCHAR aAddr, UCHAR aBytes, PUCHAR pBuf);

Функция записи массива чисел в последовательно расположенные регистры контроллера.

*Параметры:*
**Handle** – ссылка на открытый канал.
**aAddr** – адрес начального регистра контроллера.
**aBytes** – количество записываемых регистров
**pBuf** – указатель на массив значений.
*Возвращаемое значение:*
не 0 – успешное выполнение
0 – ошибка
*Примечание:*
Некоторые регистры пишутся (и читаются) только в конфигурационном режиме контроллера (см документацию на контроллер).

### ULONG CnGetRegisters(ULONG Handle, UCHAR aAddr, UCHAR aBytes, PUCHAR pBuf);

Функция чтения последовательно расположенных регистров контроллера.

*Параметры:*
**Handle** – ссылка на открытый канал.
**aAddr** – адрес регистра контроллера.
**aBytes** – количество читаемых регистров
**pBuf** – указатель на массив для сохранения значений
*Возвращаемое значение:*
не 0 – успешное выполнение
0 – ошибка

## *Работа с библиотекой (пример).*

Каждая программа, работающая с библиотекой, должна включать в себя заголовочный файл **slmcp.h**, который содержит все необходимые объявления функций и констант.

Типичная последовательность работы с библиотекой должна проводиться следующим образом:

- загрузка библиотеки ( CnOpenLibrary)
- открытие канала (CnOpen)
- инициализация ( CnInit)
- настройка скорости канала, фильтра пакетов, режима приёма пакетов (CnSetBaud, CnSetFilterMask)
- разрешение приёма пакетов (CnRecieveEnable)

- передача пакетов (CnSendPackets)
- приём пакетов (CnRecievePackets)
- запрет приёма пакетов (CnRecieveDisable)
- закрытие канала (CnClose)
- закрытие библиотеки (CnCloseLibrary)

### Текст программы «simple.cpp» для VC 6.0:

```
#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include "slmcp.h"   /* includes declarations to work with slmcp.dll */

int main(int argc, char **argv)
{
  UCHAR numChannel;    // number of using channel
  ULONG devHandle;     // hanle of device
  UCHAR devinfo;       // information of CAN controller
  bool blockread;      // mode of data read
  TMASKORFILTER mf[4]; // buffer for set receive mode, mask and filters
  TCANDATA canpacket;  //
  UCHAR status;        // status of send packet transaction
  ULONG packets;       // packet received
  UCHAR s[40];         // buffer for converter serial number
  ULONG f;

// load and open library
  if ( !CnOpenLibrary())
  { printf( "->Error of open library!");
    exit( 1);
  }

// open channel
  numChannel = 1;              // 1-255 only
  devHandle = CnOpen( numChannel);
  if ( devHandle == 0 )
  { printf( "->Error open of channel!\n");
    exit( 1);
  }

// initialization of converter
  blockread = false; //no block mode
  f = CnInit( devHandle, &devinfo, blockread);
  if ( !f )
  { printf( "->Error initialization of device command\n");
    exit( 2);
  }
  else
    if ((devinfo & 0xF0) == infoMCP2515)
      printf( "->CAN chip is MCP2515\n");

// Get converter serial number
  if ( ! CnGetSerialNumber( devHandle, &s[0], sizeof( s)))
    printf("-> Error of get serial number command\n");
  else
    printf("-> serial number of converter is %s\n", s);

// set baudrate to 500Kbit
  f = CnSetBaudRate( devHandle, cCAN_BD_500);
  if ( !f)
```

```
    printf( "->Error set baudrate command\n");

// set recieve mode, mask and filters
  mf[0].CnlInfo.NumChannel = 0; // always
  // allow filters for packets with standart and extended identifier
  mf[0].CnlInfo.IDRXMode = rxm_FILTER_ID_EXT |rxm_FILTER_ID_STD;
  mf[1].maskfilter = 0x200001; // use 0 bits for standart,
                               // and 0 and 18 bit for extended identifiers
  mf[2].maskfilter = 0x200000; // set filter 0 bit 0 for standart identifier
  mf[3].maskfilter = 0x80001;  // set filter 1 bit 0 for extended identifier
  if ( !CnSetMaskFilters( devHandle, (PULONG)&mf))
    printf("-> Error set filter and mask command\n");

// recieve packes enable
  if( ! CnRecieveEnable( devHandle))
    printf("-> Error of receive enable command\n");

// send packet
  canpacket.ID = 0x80002; // set identifier (extended)
  canpacket.DLC = 2;      // 2 bytes in data
  canpacket.d[0] = 1;
  canpacket.d[1] = 2;
  packets = sizeof( canpacket) / sizeof( TCANDATA);   // size of CAN packets buffer
  if ( ! CnSendPackets( devHandle, &canpacket, 1, &status))
    printf("-> Error of send packet command\n");
  if ( status & stat_TX_TIMEOUT)
    printf("-> TimeOut of send packet\n");

// recieve packet
  packets = sizeof( canpacket) / sizeof( TCANDATA);   // size of CAN packets buffer
  if ( ! CnRecievePackets( devHandle, &canpacket, &packets))
    printf("-> Error of receive packet command\n");
  else
    printf("-> Recieve %-.4d CAN packets\n", packets);

// recieve packes disable
  if( ! CnRecieveEnable( devHandle))
    printf("-> Error of receive disable command\n");

// Close  channel
  if( ! CnClose( devHandle))
    printf("-> Error close channel\n");

// close library
  CnCloseLibrary();
  return 0;
}
```