ООО МикроКОР **www.microkor.biz**

Карта CAN-сети

Руководство по эксплуатации

МК2.000.101 РЭ

г.Санкт-Петербург
2000 г.

Содержание

1		Основная информация	5
	1.1	Назначение	5
	1.2.	Основные параметры	5
	1.3.	Спецификация	5
2		Аппаратное обеспечение	6
	2.1.	Описание конструкции	6
	2.2.	Описание разъемов	6
	2.3.	Выбор базового адреса	7
	2.4.	Выбор прерываний	7
3		Программное обеспечение	8
	3.1.	Организация доступа к контроллеру	8
	3.2.	Программные средства mK_CAN	9
	3.3.	Программные средства mK_CAN для MS-DOS	9
	3.4.	Программные средства mK_CAN дляWindows 95/98	11

1. Основная информация

1.1. Назначение

CAN (Controller Area Network) интерфейс является разновидностью гибкого программируемого сетевого интерфейса с дифференциальным приемником-передатчиком, позволяющим формировать распределенные мультипроцессорные системы с максимальной скоростью передачи данных до 1 Mbaud , при этом обеспечивается передача до 2048 сообщений размерностью до 8 байт каждое. Максимальная скорость передачи обеспечивается при удаленности устройств на расстояние не более 40 метров.

CAN протокол специально разработан для удовлетворения требованиям, предъявляемым к сетям, используемым в промышленных условиях. Наличие дифференциальных приемо-передатчиков обеспечивает высокую помехозащищенность.

Карта CAN-сети mK_CAN предназначена для установки в персональный компьютер типа IBM PC в слот шины ISA.

1.2. Основные параметры

Карта CAN-сети обеспечивает:

- Работу одного или двух CAN-контроллеров Siemens C81C91 одновременно;
- Индивидуальную гальваническую развязку для обоих каналов связи;
- Выбор одного из 4 базовых адресов ввода-вывода;
- Выбор одной из 6 линий прерывания IRQ;

1.3. Спецификация

Таблица 1

Технические данные	
CAN контроллер	Siemens SAE 81C91 (1 или 2)
Тактовая частота	20.00 МГц
Драйвер CAN-линии	Philips 82C250 (1 или 2)
Гальваническая изоляция CAN-линии	1.5кВ; встроенный DC/DC модуль (1 или 2)
Интерфейс PC	ISA 8-bit и 16-bit
Размеры	Для установки в стандартный слот ISA 135 мм
Потребляемая мощность	100мА @ 5В
Разъем шины CAN	9-pin SUB-D
Доступные IRQ	5,7,10,11,12,15
Базовые адреса	120h, 1B0h, 220h, 300h

Обозначение при заказе:

1. Карта CAN-сети mK_CAN -X- карта с двумя CAN контроллерами (61мм)
2. Карта CAN-сети mK_CAN-03 -X карта с одним CAN контроллером (122мм)
   - **C** - общепромышленный темп. диапазон 0…+50°C);
   - **I** - индустриальный темп. диапазон (-40…+60°C).

2. Аппаратное обеспечение

2.1. Описание конструкции

Конструктивно Карта CAN-сети **mK_CAN** представляет собой печатную плату , предназначенную для установки в стандартный слот ISA.

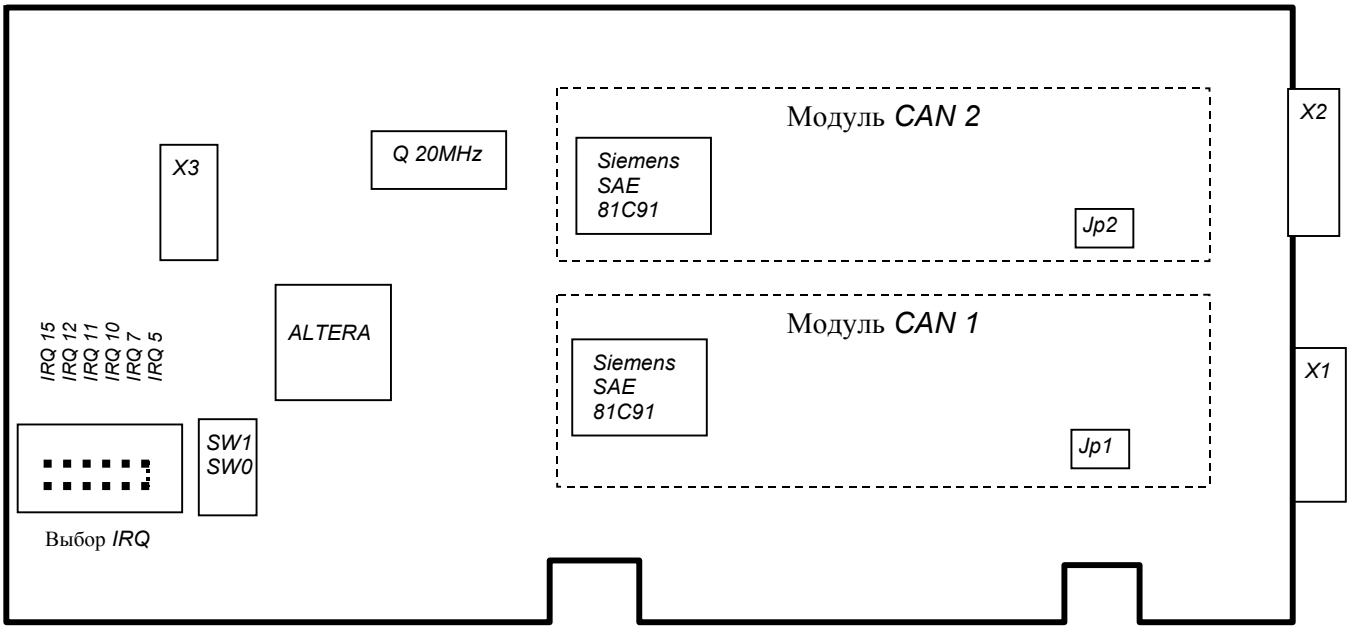


Рис. 1 Эскиз расположения на плате основных элементов

2.2. Описание разъемов

X1 – разъём канала CAN 1

В спецификации CAN подключение контроллеров к линии через разъемы не оговорено. Карта CAN сети **mK_CAN** для подключения к внешним цепям содержит разъем SUB-D (вилка) на 9 контактов. Подключение выводов разъема соответствует стандарту группы пользователей CAN " CAN in Automation" (CiA) (см. таблицу 2).

Таблица 2

Контакт	Цепь	Примечание
1	-	
2	CAN_L	Линия шины
3	GND	Общий сигн.
4	-	
5	-	
6	GND	Общий пит.
7	CAN_H	Линия шины
8	-	
9	VCC	+5В

X2 – разъём канала CAN 2

В случае использования карты CAN сети с двумя контроллерами дополнительный разъем устанавливается на место свободного слота PC и сединяется с платой гибким шлейфом с разъемом X2 на плате. (см. таблицу 3).

Таблица 3

Контакт	Цепь	Примечание
1	-	
2	GND	Общий пит.
3	CAN_L	Линия шины
4	CAN_H	Линия шины
5	GND	Общий сигн.
6	-	
7	-	
8	VCC	+5В
9	-	
10	-	

X3 – не используется.

Jp1 и Jp2 - перемычки предназначены для подключения терминатора CAN сети. Должны быть установлены, если соответствующий контроллер является оконечным устройством сети.

2.3. Выбор базового адреса

Выбор базового адреса для доступа к контроллерам CAN производится установкой переключателей SW0 и SW1в соответствующее положение (см. таблицу 4)

Таблица 4

Положение переключателяSW 1	**Положение переключателя SW0**	**Адресное пространство CAN**	**Базовый адрес БА**
ON	ON	120h…127 h	120h
ON	OFF	1B0h…1B7 h	1B0h
OFF	ON	220h…227 h	220h
OFF	OFF	300h…327 h	300h

2.4. Выбор прерываний

Выбор прерываний производится установкой перемычки на требуемый вектор прерывания. Доступны варианты: IRQ5, IRQ7, IRQ10, IRQ11, IRQ12, IRQ15, нет IRQ.

Установка двух и более перемычек выбора IRQ не рекомендуется.

3. Программное обеспечение

3.1. Организация доступа к контроллеру

Для непосредственного доступа к **CAN-контроллеру**, расположенному на плате предусмотрены семь 8-разрядных портов ввода\вывода, обеспечивающих запись и чтение регистров контроллера его аппаратный сброс и определение источника прерываний. (см.таблицу 5)

Таблица 5

Адрес	Наименование	Доступ
БА+0	Адрес регистра в CAN 1	Только запись
БА+1	Данные регистра в CAN 1	Чтение и запись
БА+2	Сброс в CAN 1	Только запись
БА+3	Адрес регистра в CAN 2	Только запись
БА+4	Данные регистра в CAN 2	Чтение и запись
БА+5	Сброс в CAN 2	Только запись
БА+6	Информация о прерываниях	Только чтение

Для обращения к регистру в CAN-контроллере (например, к CAN 1), необходимо по адресу: **Базовый Адрес+0** записать адрес этого регистра в кристалле CAN , а затем по адресу: **Базовый Адрес+1** считать или записать данные.

Сброс CAN-кристалла осуществляется посредством последовательной записи по адресу **Базовый Адрес+2 (Базовый Адрес+5)** числа 00h и через 100 мс -числа 01h.

В случае возникновения прерывания от одного из кристаллов CAN или от обоих одновременно по адресу **Базовый Адрес+6** можно установить источник прерывания. При наличии только одного кристалла в этом нет необходимости.

Для установления источника прерывания необходимо считать байт по адресу **Базовый Адрес+6.**
В случае, если бит 0=1 – произошло прерывание от кристалла CAN 1.
В случае, если бит 1=1 – произошло прерывание от кристалла CAN 2.

3.2. Программные средства mK_CAN

Плата **mK_CAN** поставляется с драйверами и демонстрационными программами для операционных систем Microsoft Windows 95OSR2, 98, Windows NT 4.0, а также MS-DOS версий 6.2 и более поздних. Для консультаций обращайтесь по электронной почте e-mail по адресу can@microkor.biz.

Программные средства для ОС MS-DOS являются полнофункциональными. Для ОС Windows поставляется демо-версия программного обеспечения, обеспечивающая максимальную скорость передачи 2500 сообщений в секунду.

Программное обеспечение, а также демонстрационные программы устанавливаются на персональный компьютер с помощью инсталлятора (mK_CANinstall.exe), поставляемого на дискете в комплекте с картой **mK_CAN**.

3.3. Программные средства mK_CAN для MS-DOS

Резидентный драйвер mKCANDOS.EXE. предназначен для управления CAN контроллером SAE81C91, входящим в состав карты расширения **mK_CAN** для PC, из прикладных программ. Копирование драйвера на жесткий диск PC производится автоматически с помощью инсталлятора (mK_CANinstall.exe) если установлен флаг «MS-DOS stuff».

Для установки драйвера наберите в среде MS-DOS командную строку:

[d:][path]mKCANDOS.EXE /B=base [/I=IRQ] /S=serv

и нажмите Enter. Установка драйвера может также осуществляться с помощью командного файла (например autoexec.bat). Пример установки драйвера с помощью командного файла (mkcandos.bat) поставляется вместе с драйвером.

Параметры, используемые при установке:

d: диск, на котором находится файл драйвера,
path путь к файлу драйвера,
base базовый адрес карты **mK_CAN**,
IRQ номер линии прерывания, используемый картой **mK_CAN**
(если параметр не задан, прерывания от CAN контроллера не используются),
serv номер вектора прерывания для установки обслуживающих функций драйвера.
Для предотвращения конфликтов выбирайте номер неиспользуемого вектора.

Если строка параметров содержит ошибки, то выводится сообщение об ошибке, драйвер не устанавливается.

При успешной установке драйвер выводит на экран логотип и сообщение об установке. Драйвер производит аппаратный сброс обоих контроллеров CAN и инициализирует CC (Clock Control) регистр первого контроллера CAN в 0, устанавливая тем самым тактовую частоту для второго контроллера CAN такой же, как для первого (20 МГц).

Если установлены несколько карт **mK_CAN**, то для каждой требуется отдельная установка драйвера с соответствующими параметрами (при этом все их параметры base, IRQ, serv должны быть различны).

Драйвер предоставляет ряд сервисных функций. Вызов этих функций осуществляется через обслуживающее прерывание, номер которого настраивается при запуске драйвера, а параметры передаются через регистры процессора в соответствии со следующими правилами:

AH на входе: номер функции
AL на входе: адрес байта
BL на входе, выходе: значение байта
ES:BX на входе, выходе: адрес (сегмент:смещение) структуры из 8 байтов, содержащей данные сообщения либо адрес функции реакции на прерывание.
AX на выходе: 0, если функция выполнена успешно, иначе - код ошибки.

Коды ошибок, возвращаемых драйвером в регистре AX:

0: успешное выполнение (нет ошибки).
1: одновременно более 1 обращения к драйверу, запрос не выполнен.
2: неверный номер функции в регистре AH (должен быть от 0 до 5), запрос не выполнен.
3: задан адрес байта в области сообщений, запрос выполнен, но корректность данных не гарантируется. Правильную последовательность обмена должна в этом случае обеспечить вызывающая программа.Рекомендуется использовать для области сообщений соответствующие функции драйвера.
4: неверный номер сообщения в регистре AL (должен быть от 0 до 15), запрос не выполнен.

Обслуживающие функции драйвера (указаны регистры на входе).

AH=0: аппаратный сброс контроллера CAN 1.
После этого для установки тактовой частоты для второго контроллера CAN такой же, как для первого, требуется задать CC (Clock Control) регистр (адрес 0x14) первого контроллера CAN равным 0 (записать 0x80, затем 0x00).
AH=1: установка функции реакции на прерывание (типа far) для обоих контроллеров CAN, ES:BX = адрес функции, CX = регистр DS для функции.
AH=2: чтение байта с адреса AL контроллера CAN 1 в BL.
AH=3: запись байта из BL по адресу AL контроллера CAN 1.
AH=4: чтение 8 байтов из сообщения с номером AL (0..15) контроллера CAN 1 в структуру по адресу ES:BX.
AH=5: запись 8 байтов из структуры по адресу ES:BX в сообщение с номером AL (0..15) контроллера CAN 1.
AH=6: аппаратный сброс контроллера CAN 2.
AH=7: установка функции реакции на прерывание (типа far) для обоих контроллеров CAN, ES:BX = адрес функции, CX = регистр DS для функции.
AH=8: чтение байта с адреса AL контроллера CAN 2 в BL.
AH=9: запись байта из BL по адресу AL контроллера CAN 2.
AH=10: чтение 8 байтов из сообщения с номером AL (0..15) контроллера CAN 2 в структуру по адресу ES:BX.
AH=11: запись 8 байтов из структуры по адресу ES:BX в сообщение с номером AL (0..15) контроллера CAN 2.

Описание функции реакции на прерывание.

Функция реакции на прерывание устанавливается для обоих контроллеров CAN.

Перед окончанием программы должен быть задан адрес 0:0 для запрета вызовов функции. Реакция на прерывание осуществляется только если при установке драйвера был задан параметр IRQ.

Если прерывание произошло во время выполнения вызова драйвера, то функция реакции на прерывание будет вызвана по окончании выполнения текущего вызова.

При вызове функции реакции на прерывание драйвер обеспечивает сохранение и восстановление всех регистров процессора (80286), а также окружения (PSP, который был при установке функции).

Если DS не был задан при установке, то для доступа к данным функция реакции на прерывание должна в начале настроить регистр DS.

Функция реакции на прерывание должна быть короткой по времени, например, установка флага прерывания.

Описание функции обработки прерывания.

Функция обработки прерывания в фоновой программе опрашивает в цикле флаг прерывания и, если он установлен, должна выполнить опрос и обработку обоих контроллеров CAN.

При этом для разрешения генерации следующих прерываний должен быть выполнен сброс сигналов прерываний от обоих контроллеров CAN: например, обнуление регистров IMSK обоих контроллеров CAN (после обработки данных и обнуления флагов прерываний регистры IMSK должны быть восстановлены для генерации следующих прерываний)

Пример использования драйвера в прикладной программе прилагается в виде исходного текста на языке C++ (drv_test.cpp) и исполняемой программы (drv_test.exe). Для запуска теста можно использовать командный файл drv_test.bat.

3.4. Программные средства mK_CAN для Windows 95/98

В комплекте с платой **mK_CAN** поставляется виртуальный драйвер (Vichw11.vxd) платы для операционной системы Windows 95\98 и соответствующая динамическая библиотека (mKCAN.dll) управления драйвером.

Прилагаются также тестовая программа (TestCanDll.exe) и пример программирования (CAN_PC) на языке Object Pascal в среде Borland Delphi 3.0 и старше.

Установка драйвера и динамической библиотеки производится автоматически с помощью инсталлятора (setup.exe).

Экспортируемые функции драйвера описаны в файле CANdll.pas, прилагаемом с примером. Типы используемых данных описаны в файле CANdefine.pas.

Для вызова функций используется соглашение stdcall.

Краткое описание функций.

function **_OpenDriver**(BASE:word):boolean;

Назначение	Запуск драйвера. Необходимо для начала работы.
Аргументы	WORD базовый адрес ввода-вывода
Результаты	BOOLEAN True – драйвер запущен False – драйвер не запущен

procedure **_CloseDriver;**

Назначение	Останов драйвера. Необходимо для корректного завершения работы.
Аргументы	Нет
Результаты	Нет

function **_RdCAN**(ncontr: byte; Addr : word) : Byte;

Назначение	Чтение регистров контроллера CAN.
Аргументы	BYTE номер контроллера WORD адрес регистра
Результаты	BYTE значение регистра
!	Для чтения памяти сообщений пользуйтесь функцией _ReadMessage

procedure **_WrCAN**(ncontr: byte; Addr : word; nNewValue : Byte);

Назначение	Запись регистров контроллера CAN.
Аргументы	BYTE номер контроллера WORD адрес регистра BYTE новое значение
Результаты	Нет
!	Для записи памяти сообщений пользуйтесь функцией _SetupMessage

function **_ResetCAN**(ncontr: byte):boolean;

Назначение	Аппаратный сброс контроллера.
Аргументы	BYTE номер контроллера
Результаты	BOOLEAN True – сброс выполнен False – сброс не выполнен

function **_ActiveHW**:boolean;

Назначение	Тест драйвера.
Аргументы	Нет
Результаты	BOOLEAN True – драйвер функционирует нормально False – драйвер не запущен или произошёл внутренний сбой

function **_GetIRQNumber**:byte;

Назначение	Получить номер установленного IRQ
Аргументы	Нет
Результаты	BYTE Номер установленного IRQ

procedure **_SetIRQNumber**(nNewValue : Byte);

Назначение	Назначить номер IRQ
Аргументы	BYTE номер IRQ
Результаты	Нет
!	Используйте функцию _SetIRQ

procedure **_SetHardAccess**(parm : Boolean);

Назначение	Установка режима доступа к оборудованию
Аргументы	BOOLEAN True – доступ через функции ядра (рекомендуется) False – непосредственный доступ
Результаты	Нет

function **_GetHardAccess**:Boolean;

Назначение	Определить текущий режим доступа к оборудованию
Аргументы	Нет
Результаты	BOOLEAN True – доступ через функции ядра (рекомендуется) False – непосредственный доступ

procedure **_SetIRQMasked**(bNewValue : Boolean);

Назначение	Установка маски IRQ
Аргументы	BOOLEAN True – маска установлена False – маска снята
Результаты	Нет

procedure **_SetIRQ**(IRQNumber : Word; HWHandler : TInterruptHandler):Boolean;

Назначение	Установка обработчика прерывания
Аргументы	BYTE номер IRQ TInterruptHandler обработчик прерывания
Результаты	BOOLEAN True – обработчик прерывания установлен False – обработчик прерывания не установлен

function **GetCANCB**(ncontr: byte; pCB:PCANCB):byte;

Назначение	Получение CAN Control Block
Аргументы	BYTE номер контроллера PCANCB указатель на структуру CAN Control Block
Результаты	Код ошибки: CANCB_OK – нет ошибки CANCB_ERROR – неопознанная ошибка CANCB_TSEG1_OUT_OF_RANGE CANCB_TSEG2_OUT_OF_RANGE CANCB_SJW_OUT_OF_RANG CANCB_TSP_OUT_OF_RANGE CANCB_BRP_OUT_OF_RANGE CANCB_CC_OUT_OF_RANGE CANCB_TCEC_OUT_OF_RANGE

function **SetCANCB**(ncontr: byte; pCB:PCANCB):byte;

Назначение	Запись CAN Control Block в CAN – контроллер
Аргументы	BYTE Номер контроллера PCANCB указатель на структуру CAN Control Block
Результаты	Код ошибки: CANCB_OK – нет ошибки CANCB_ERROR – неопознанная ошибка CANCB_TSEG1_OUT_OF_RANGE CANCB_TSEG2_OUT_OF_RANGE CANCB_SJW_OUT_OF_RANG CANCB_TSP_OUT_OF_RANGE CANCB_BRP_OUT_OF_RANGE CANCB_CC_OUT_OF_RANGE CANCB_TCEC_OUT_OF_RANGE

function **ClearCANCB**(ncontr: byte;pCB:PCANCB):byte;

Назначение	Очистка структуры CAN Control Block
Аргументы	BYTE номер контроллера PCANCB указатель на структуру CAN Control Block
Результаты	Код ошибки: CANCB_OK – нет ошибки CANCB_ERROR – неопознанная ошибка CANCB_TSEG1_OUT_OF_RANGE CANCB_TSEG2_OUT_OF_RANGE CANCB_SJW_OUT_OF_RANG CANCB_TSP_OUT_OF_RANGE CANCB_BRP_OUT_OF_RANGE CANCB_CC_OUT_OF_RANGE CANCB_TCEC_OUT_OF_RANGE

function **GetRR**(ncontr: byte):TrRR;

Назначение	Получение регистров Receive Ready
Аргументы	BYTE номер контроллера
Результаты	TrRR слово состояния регистров RRx

procedure **SetRR**(ncontr: byte; RR:TrRR);

Назначение	Запись регистров Receive Ready
Аргументы	BYTE номер контроллера TrRR слово состояния регистров RRx
Результаты	Нет

function **GetRIM**(ncontr: byte):TrRIM;

Назначение	Получение регистров Receive Interrupt Mask
Аргументы	BYTE номер контроллера
Результаты	TrRIM слово состояния регистров RIMx

procedure **SetRIM**(ncontr: byte; RR:TrRIM);

Назначение	Запись регистров Receive Interrupt Mask
Аргументы	BYTE номер контроллера TrRIM слово состояния регистров RIMx
Результаты	Нет

function **GetTRS** (ncontr: byte):TrTRS;

Назначение	Получение регистров Transmit Request Set
Аргументы	BYTE номер контроллера
Результаты	TrTRS слово состояния регистров TRSx

procedure **SetTRS**(ncontr: byte;RR:TrTRS);

Назначение	Запись регистров Transmit Request Set
Аргументы	BYTE номер контроллера TrTRS слово состояния регистров TRSx
Результаты	Нет

procedure **SetTRR**(ncontr: byte;RR:TrTRR);

Назначение	Запись регистров Transmit Request Reset
Аргументы	BYTE номер контроллера TrTRR слово состояния регистров TRRx
Результаты	Нет

function **GetRRP**(ncontr: byte):TrRRP;

Назначение	Получение регистров Receive Request Pending
Аргументы	BYTE номер контроллера
Результаты	TrRRP слово состояния регистров RRPx

procedure **EnterSetup**(ncontr: byte);

Назначение	Выполнить программный сброс и открыть доступ к защищённым регистрам CAN – контроллера
Аргументы	BYTE номер контроллера
Результаты	Нет

procedure **ExitSetup**(ncontr: byte);

Назначение	Закрыть доступ к защищённым регистрам CAN – контроллера и перевести контроллер в нормальный режим работы.
Аргументы	BYTE номер контроллера
Результаты	Нет

function **SetupMessage**(ncontr: byte;pMSG:PCANMSG; box:byte):byte;

Назначение	Запись сообщения в память CAN – контроллера
Аргументы	BYTE номер контроллера PCANMSG указатель на сообщение CAN BYTE номер объекта в памяти сообщений
Результаты	Код ошибки CANCB_OK – нет ошибки CANCB_ERROR – неопознанная ошибка CANCB_DLC_OUT_OF_RANGE CANCB_ID_OUT_OF_RANGE CANCB_BOX_OUT_OF_RANGE

function **ReadMessage**(ncontr: byte;pMsg:PCANMSG; box:byte);

Назначение	Чтение сообщения из памяти CAN – контроллера
Аргументы	BYTE номер контроллера PCANMSG указатель на сообщение CAN BYTE номер объекта в памяти сообщений
Результаты	Код ошибки CANCB_OK – нет ошибки CANCB_ERROR – неопознанная ошибка CANCB_BOX_OUT_OF_RANGE

function **GetInterruptSource**:Byte;

Назначение	Получение источника прерывания
Аргументы	Нет
Результаты	BYTE номер контроллера

Типы используемых данных:

```
TInterruptHandler = procedure (Sender       : TObject;
                               HwCounter    : Longint); far stdcall;
```

Sender – объект, вызвавший прерывание

HwCounter – счётчик зарегистрированных прерываний с момента перезапуска драйвера

```
TCANMSG = record                        - сообщение CAN
        ID:word;                        - идентификатор сообщения (0..2047)
        RTR:boolean;                    - True: бит RTR=1 False: бит RTR:=0 [1]
        DLC:byte;                       - длина сообщения (0..8)
        MSGbody:array[0..7] of byte;    - тело сообщения
        TimeStamp:word;                 - метка времени
    end;
PCANMSG=^TCANMSG;

TrBL1 = record                          - структура для хранения регистра BL1
        TSEG1:byte;
        TSEG2:byte;
        SAM:boolean;
    end;

TrBL2 = record                          - структура для хранения регистра BL2
        SJW:byte;
        SM:boolean;
        DI:boolean;
        IPOL:boolean;
    end;

TrOC  = record                          - структура для хранения регистра OC
        OCTP1:boolean;
        OCTN1:boolean;
        OCP1:boolean;
        OCTP0:boolean;
        OCTN0:boolean;
        OCP0:boolean;
        OCM1:boolean;
        OCM0:boolean;
    end;

TrRR  = word;                           - структура для хранения регистров RRx

TrRIM = word;                           - структура для хранения регистров RIMx

TrTRS = word;                           - структура для хранения регистров TRSx

TrIMSK = record                         - структура для хранения регистра IMSK
```

[1] Здесь и везде используется следующее соглашение для хранения битовых данных:
True: бит установлен (1)
False: бит не установлен (0)

```
        ERI:boolean;
        ETI:boolean;
        EWLI:boolean;
        ERFI:boolean;
        EWUPI:boolean;
        EBOI:boolean;
        EEPI:boolean;
        ETCI:boolean;
    end;

TrMOD = record              - структура для хранения регистра MOD
        IM:boolean;
        RES:boolean;
        BS:boolean;
        RWL:boolean;
        TWL:boolean;
        TC:boolean;
        RS:boolean;
        ADE:boolean;
   end;

TrINT = record              - структура для хранения регистра INT
        RI:boolean;
        TI:boolean;
        WLI:boolean;
        RFI:boolean;
        WUPI:boolean;
        BOI:boolean;
        EPI:boolean;
        TCI:boolean;
    end;

TrCTRL = record                 - структура для хранения регистра CTRL
        MM:boolean;
        TCE:boolean;
        SME:boolean;
        TSOV:boolean;
        TSP:byte;
        TST:boolean;
        RX:boolean;
    end;

 TrTRR = word;                  - структура для хранения регистров TRRx

 TrRRP = word;                      - структура для хранения регистров RRPx


   type TCANCB = record             - структура для хранения CAN Control Block
    rBL1:TrBL1;
    rBL2:TrBL2;
    rOC:TrOC;
    rBRP:byte;
    rIMSK:TrIMSK;
```

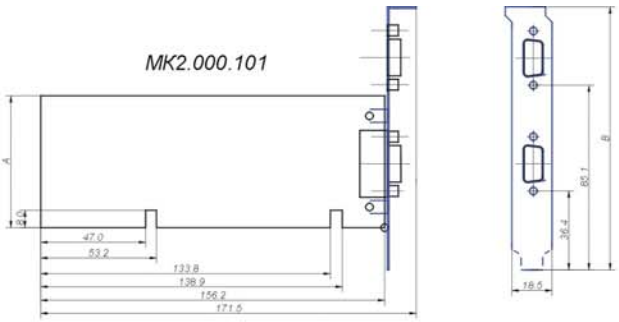
```
        rMOD:TrMOD;
        rINT:TrINT;
        rCTRL:TrCTRL;
        rCC:byte;
        rTCEC:byte;
        rTSC:word;
        drvERR:word;                    - код внутренней ошибки драйвера (не
                                          предназначен для использования)

    end;
     PCANCB=^TCANCB;
```

Примерный сценарий инициализации карты mK_CAN:

```
procedure DemoHandler(Sender : TObject; HwCounter : Longint); stdcall;
begin
                                        - процедура обработки прерываний
end;

var
     MyCAN:TCANCB;
     MyMSG:TCANMSG;

procedure InitCAN;

  var Base,IRQ:word;
      CANerror,i:byte;
      pMyCAN:PCANCB;
      pMyMSG:PCANMSG;
      contr:byte;
  begin

              contr:=0;                 - выбор контроллера
              pMyCAN:=@MyCAN;
              pMyMSG:=@MyMSG;
              ClearCANCB(contr,pMyCAN);       - очистка CAN control block
              Base:=$1B0;               - выбор базового адреса
              IRQ:=10;                  - выбор линии IRQ
              if not OpenDriver(Base) then    - запуск драйвера
              begin
                    ShowMessage('Driver not opened');
                  exit;
               end;
            if not ActiveHW then        - тест драйвера
               begin
                    ShowMessage ('Driver not started');
                    exit;
               end;
              ResetCAN(contr);                - аппаратный сброс карты CAN
              if GetCANCB(contr,pMyCAN)<>CANCB_OK then – чтение CAN control block
```

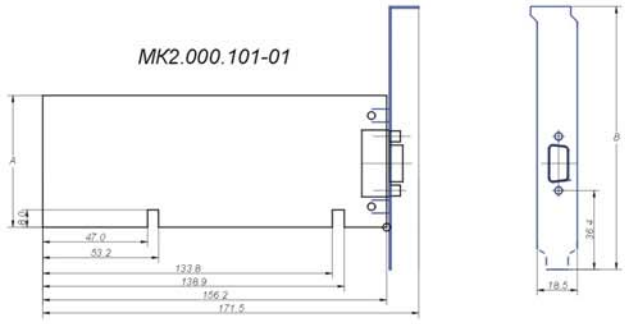
```
    begin
        ShowMessage ('Can not get CAN control block');
        exit;
    end;
  with MyCAN do                    - инициализация регистров CAN
    begin
      rBRP:=$0A;
      rBL1.TSEG1:=$03;
      rBL1.TSEG2:=$02;
      rBL2.DI:=True;
      rBL2.SJW:=$01;
      rIMSK.ERI:=True;
      rOC.OCTN0:=True;
      rOC.OCTP0:=True;
      rCC:=CC_f2;
      rCTRL.MM:=True;
    end;
  EnterSetup(contr);                          - разрешение доступа к регистрам CAN
    CANError:=SetCANCB(contr,pMyCAN);         - запись CAN control block
    if CANError<>CANCB_OK then
    begin
        ShowMessage ('Can not get CAN control block');
        exit;
    end;
    MyMSG.ID:=0;                  - подготовка "чистого" сообщения
    MyMSG.RTR:=false;
    MyMSG.DLC:=8;
    MyMSG.MSGbody[0]:=$FF;
    MyMSG.MSGbody[1]:=$FF;
    MyMSG.MSGbody[2]:=$FF;
    MyMSG.MSGbody[3]:=$FF;
    MyMSG.MSGbody[4]:=$FF;
    MyMSG.MSGbody[5]:=$FF;
    MyMSG.MSGbody[6]:=$FF;
    MyMSG.MSGbody[7]:=$FF;
    for i:=0 to 15 do SetupMessage(pMyMSG,i);
                                  - принудительная очистка памяти сообщений
                                  CAN
  SetIRQ(IRQ,DemoHandler);        - установка обработчика прерываний
    ExitSetup(contr);                         - завершение инициализации
 SetIRQMasked(false);             - разрешение обработки прерываний
end;
```

МК2.000.101

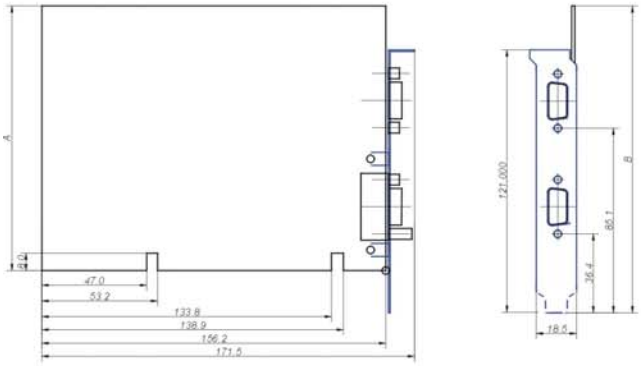


Обозначение	Размер		Масса
	A	B	
МК2.000.101	60.5	121.2	79г.
МК2.000.101-01	60.5	121.2	64г.
МК2.000.101-02	122.0	141.5	104г.
МК2.000.101-03	122.0	141.5	89г.

МК2.000.101-01



МК2.000.101-02



МК2.000.101-03