

**Справочное описание  
интерфейса драйверов  
tmkll4.h v7.xx (DOS),  
ntmk.c v4.xx (Windows NT 4.0),  
WDMTMKv2.cpp v4.xx (Windows 98/ME/2000/XP/Vista/7),  
ltmk.c v4.xx (Linux kernel 2.2/2.4/2.6/3.0),  
qnx6tmk.c v4.xx (QNX 6.x).**

Применимость описанных компонентов драйвера указывается перечислением ОС в квадратных скобках в заголовке описания компонента:

[DOS] - применимость в DOS,

[Win] - применимость в Windows NT 4.0 и в Windows 98/ME/2000/XP/Vista/7,

[Linux] - применимость в Linux,

[QNX6] - применимость в QNX 6.

Для универсальных компонентов драйвера (одинаковых для всех драйверов) применимость [DOS, Win, Linux, QNX6] не указывается.

Все универсальные функции описаны в формате драйвера DOS, под unsigned подразумевается 16 бит без знака, под unsigned long подразумевается 32 бита без знака, под int подразумевается в DOS 16 бит со знаком, в Windows, Linux, QNX6 - 32 бита со знаком.

\*\*\*\*\*

### Переменная Кода Ошибки

\*\*\*\*\*

#### tmkError

```
int tmkError;
```

При возникновении ошибочной ситуации в tmkError содержится код ошибки, обнаруженной драйвером:

TMK\_BAD\_TYPE - недопустимый тип устройства ТМК;

TMK\_BAD\_IRQ - недопустимый номер линии запроса прерывания;

TMK\_BAD\_NUMBER - недопустимый номер устройства ТМК;

TMK\_BAD\_FUNC - данный тип устройства не поддерживает вызванную функцию;

TMK\_PCI\_ERROR - неправильная попытка сконфигурировать устройство PCI;

BC\_BAD\_BUS - для КК без резервирования задана резервная ЛПИ;

BC\_BAD\_BASE - недопустимая база ДОЗУ КК;

BC\_BAD\_LENGTH - недопустимая длина блока слов для КК;

BC\_BAD\_FUNC - данный тип устройства КК не поддерживает вызванную функцию.

RT\_BAD\_PAGE - недопустимая страница ДОЗУ ОУ;

RT\_BAD\_LENGTH - недопустимая длина блока слов для ОУ;

RT\_BAD\_ADDRESS - недопустимый адрес ОУ;

RT\_BAD\_FUNC - данный тип устройства ОУ не поддерживает вызванную функцию.

\*\*\*\*\*

### Типы Прерываний

\*\*\*\*\*

#### Прерывания КК

bcIntNorm - прерывание нормального завершения одиночного обмена КК (после bcstart).

Данные: wResult == 0 (слово результата обмена равно нулю).

bcIntExc - прерывание завершения одиночного обмена КК (после bcstart) с исключительной ситуацией. Данные: wResult (слово результата обмена), wAW1, wAW2 (ответные слова в порядке их поступления в КК; для форматов с одним ОС второе ОС не определено; при обнаружении ошибки во время обмена (определяется по слову результата обмена) оба ОС имеют неопределенное состояние).

bcIntX - прерывание завершения обмена (останова) КК (после bcstartx). Данные: wResultX (расширенное слово результата обмена), wBase (база останова).

bcIntSig - сигнальное прерывание КК (после bcstartx). Данные: wBase (текущая база).

#### Прерывания МТ

mtIntX - прерывание останова МТ (после mtstartx). Данные: wResultX (расширенное слово результата), wBase (база останова).

mtIntSig - сигнальное прерывание МТ (после mtstartx). Данные: wBase (текущая база).

#### Прерывания ОУ

rtIntCmd - прерывание по команде режима управления ОУ. Данные: wCmd (код команды режима управления по маске 0x041F, то есть сама команда (младшие 5 бит) и бит "прием/передача" командного слова данной команды).

rtIntErr - прерывание по ошибке обмена ОУ. Данные: wStatus (слово состояния ОУ с установленным битом "Ошибка сообщения МК" и битами командного слова, при обработке которого возникла ошибка).

rtIntData - прерывание по команде приема/передачи данных ОУ. Данные: wStatus (слово состояния ОУ, содержащее младшие 11 битов КС).

#### Прерывания ТМК

tmkIntOth - дополнительное прерывание, не привязанное к режиму работы ТМК (прерывание таймера, переполнение буфера прерываний). Данные: wRequest (слово запроса прерывания, содержащее битовую маску дополнительных прерываний).

\*\*\*\*\*

### Типы Данных

\*\*\*\*\*

#### TTmkEventData [Win, Linux, QNX6]

```
typedef struct
{
    int nInt;
    unsigned short wMode;
    union
    {
        struct
        {
            unsigned short wResult;
            unsigned short wAW1;
        }
    }
}
```

```

        unsigned short wAW2;
    } bc;
    struct
    {
        unsigned short wBase;
        unsigned short wResultX;
    } bcx;
    struct
    {
        unsigned short wStatus;
        unsigned short wCmd;
    } rt;
    struct
    {
        unsigned short wBase;
        unsigned short wResultX;
    } mt;
    struct
    {
        unsigned short wStatus;
    } mrt;
    struct
    {
        unsigned short wRequest;
    } tmk;

};
} TTmkEventData;

```

Драйвер передает процессу информацию о всех прерываниях через структуру типа TTmkEventData. Для получения информации используется функция tmkgetevd.

### TMK\_DATA [Win, Linux, QNX6]

```
#define TMK_DATA unsigned short
```

Тип данных 16 бит без знака, используемый для описания параметров функций драйвера.

\*\*\*\*\*

### Функции для Контроллера Канала

\*\*\*\*\*

#### bcdefbase

```
int bcdefbase(unsigned bcBasePC);
```

Параметры:

bcBasePC - устанавливаемая база.

Возвращаемое значение:

0 - функция выполнена успешно;

BC\_BAD\_BASE - заданное значение базы находится вне допустимого диапазона.

Выполняемые действия:

Функция настраивает выбранный КК и драйвер на дальнейшую работу с ДОЗУ в указанной базе. Вызов функции с недопустимым значением базы приводит к ошибочной ситуации с кодом ошибки BC\_BAD\_BASE.

#### bcdefbus

```
int bcdefbus(unsigned bcBus);
```

Параметры:

bcBus - выбираемая ЛПИ (BUS\_1 или BUS\_A для основной ЛПИ, BUS\_2 или BUS\_B для резервной ЛПИ).

Возвращаемое значение:

0 - функция выполнена успешно;

BC\_BAD\_BUS - если для устройства без резервирования задана резервная ЛПИ.

Выполняемые действия:

Функция настраивает выбранный КК на обмен по заданной ЛПИ. Весь дальнейший обмен с МК осуществляется по этой ЛПИ до следующего вызова bcdefbus или bcreset (bcreset устанавливает основную ЛПИ). Вызов функции со значением BUS\_2 или BUS\_B для устройства без резервирования приводит к ошибочной ситуации с кодом ошибки BC\_BAD\_BUS.

### **bcdefintexc [DOS]**

```
void bcdefintexc(void (far* UserExcBC)(unsigned, unsigned, unsigned));
```

Параметры:

UserExcBC - пользовательская функция, вызываемая при завершении обмена, запущенного функцией bcstart, с исключительной ситуацией (прерывание bcIntExc); функция должна быть описана как far-функция, возвращающая тип void, и получающая через параметры три слова: слово результата обмена, первое ОС, второе ОС (порядок ответных слов определяется порядком их поступления в КК); для форматов с одним ОС второе ОС не определено; при обнаружении ошибки во время обмена (определяется по слову результата обмена) оба ОС имеют неопределенное состояние.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция задает драйверу имя пользовательской функции, которая будет вызываться каждый раз при завершении обмена по МК выбранным КК с исключительной ситуацией (обнаружены ошибки или установленные биты в поле флагов ОС) (см. Руководство программиста), если обмен был запущен функцией bcstart.

### **bcdefintnorm [DOS]**

```
void bcdefintnorm(void (far* UserNormBC)(unsigned, unsigned, unsigned));
```

Параметры:

UserNormBC - пользовательская функция, вызываемая при нормальном завершении обмена, запущенного функцией bcstart (прерывание bcIntNorm); функция должна быть описана как far-функция, возвращающая тип void, и получающая через параметры три слова: слово результата обмена, всегда равное 0, и два фиктивных параметра, введенных для совместимости с UserExcBC (неопределенные первое и второе ОС).

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция задает драйверу имя пользовательской функции, которая будет вызываться каждый раз при нормальном завершении обмена по МК выбранным КК (не обнаружено ошибок или установленных бит в поле флагов ОС) (см. руководство программиста), если обмен был запущен функцией bcstart.

### **bcdefintsig [DOS]**

```
void bcdefintsig(void (far* UserSigBC)(unsigned));
```

Параметры:

UserSigBC - пользовательская функция, вызываемая при возникновении сигнального прерывания во время обмена КК, запущенного функцией bcstartx (прерывание bcIntSig); функция должна быть описана как far-функция, возвращающая тип void, и получающая через параметры одно слово: номер базы, в которой возникло сигнальное прерывание.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция задает драйверу имя пользовательской функции, которая будет вызываться каждый раз при возникновении сигнального прерывания во время обмена КК, запущенного функцией bcstartx. Если время выполнения пользовательской функции обработки сигнальных прерываний больше периода их возникновения, возможен пропуск вызовов пользовательской функции для промежуточных прерываний. Пользовательская функция должна сама определять такую ситуацию по получаемому параметру (номеру базы возникновения прерывания).

### **bcdefintx [DOS]**

```
void bcdefintx(void (far* UserXBC)(unsigned, unsigned));
```

Параметры:

UserXBC - пользовательская функция, вызываемая при завершении обмена КК, запущенного функцией bcstartx (прерывание bcIntX); функция должна быть описана как far-функция, возвращающая тип void, и получающая через параметры два слова: номер базы, в которой произошел останов, слово расширенного результата обмена.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция задает драйверу имя пользовательской функции, которая будет вызываться каждый раз при завершении обмена по МК выбранным КК, если обмен был запущен функцией bcstartx.

### **bcdefirqmode**

```
int bcdefirqmode(unsigned bcIrqMode);
```

Параметры:

bcIrqMode - битовая маска управления режимом прерываний КК:

TMK\_IRQ\_OFF - задание этого бита отключает прерывание устройств ТМКХ, ТА от шины ISA;

BC\_GENER1\_BL - задание этого бита блокирует прерывания по обнаружению генерации в основной ЛПИ устройства ТМКХ;

BC\_GENER2\_BL - задание этого бита блокирует прерывания по обнаружению генерации в резервной ЛПИ устройства ТМКХ.

Возвращаемое значение:

0 - функция выполнена успешно;

BC\_BAD\_FUNC - выбранный КК не поддерживает программное управление режимом прерываний КК.

Выполняемые действия:

Функция задает режим прерываний выбранного КК. По умолчанию всегда устанавливаются биты BC\_GENER1\_BL и BC\_GENER2\_BL, так как признак генерации может возникать асинхронно, а никакой обработки признака в драйвере нет.

Единственной правильной реакцией на возникновение признака генерации может быть сброс устройства вызовом bcreset или блокировка признаков генерации в пользовательской функции обработки прерывания. Если биты признаков генерации не анализируются в пользовательских функциях обработки прерываний, запрещается снимать блокировки прерываний по генерации.

### **bcdeflink**

```
int bcdeflink(unsigned bcBase, unsigned bcCtrlCodeX);
```

Параметры:

bcBase - база, следующая в цепочке за текущей выбранной bcdefbase;

bcCtrlCodeX - расширенный код управления для базы bcBase, объединяющий по "ИЛИ":

формат обмена:

DATA\_BC\_RT - передача данных КК-ОУ,  
DATA\_BC\_RT\_BRCST - передача данных КК-ОУ (групповая),  
DATA\_RT\_BC - передача данных ОУ-КК,  
DATA\_RT\_RT - передача данных ОУ-ОУ,  
DATA\_RT\_RT\_BRCST - передача данных ОУ-ОУ (групповая),  
CTRL\_C\_A - команда управления КС-ОС,  
CTRL\_C\_BRCST - команда управления КС (групповая),  
CTRL\_CD\_A - команда управления КС+ИС-ОС,  
CTRL\_CD\_BRCST - команда управления КС+ИС (групповая),  
CTRL\_C\_AD - команда управления КС-ОС+ИС;

признак продолжения:

CX\_CONT - по окончании обмена перейти в следующую базу цепочки, если нет ошибок или установленных битов ОС

CX\_STOP - остановиться по окончании обмена;

номер ЛПИ:

CX\_BUS\_A (или CX\_BUS\_0) - основная ЛПИ,  
CX\_BUS\_B (или CX\_BUS\_1) - резервная ЛПИ;

требование формирования сигнального прерывания:

CX\_SIG - сформировать сигнальное прерывание при старте обмена,

CX\_NOSIG - не формировать сигнальное прерывание.

Возвращаемое значение:

0 - функция выполнена успешно;

BC\_BAD\_BASE - заданное значение базы находится вне допустимого диапазона;

BC\_BAD\_FUNC - выбранный КК не поддерживает старт обмена через bcstartx и формирование цепочек.

Выполняемые действия:

Функция добавляет в цепочку после текущей выбранной базы базу bcBase выбранного КК. Если затем при старте bcstartx управление по цепочке дойдет до текущей базы и в расширенном коде управления для текущей базы будет стоять признак продолжения CX\_CONT, то следующим обменом в цепочке будет обмен из базы bcBase с кодом управления bcCtrlCodeX. Цепочка остановится, если выполнится обмен с признаком CX\_STOP в коде управления, также цепочка остановится по любой ошибке или установленному биту в принятом ОС. Вызов функции с недопустимым значением базы приводит к ошибочной ситуации с кодом ошибки BC\_BAD\_BASE. Эта функция работает на устройствах ТМКХ, ТМКХI, ТА, ТАI.

## **bcgetansw**

`unsigned long bcgetansw(unsigned bcCtrlCode);`

Параметры:

bcCtrlCode - код управления, задающий формат обмена:

DATA\_BC\_RT - передача данных КК-ОУ,  
DATA\_BC\_RT\_BRCST - передача данных КК-ОУ (групповая),  
DATA\_RT\_BC - передача данных ОУ-КК,  
DATA\_RT\_RT - передача данных ОУ-ОУ,  
DATA\_RT\_RT\_BRCST - передача данных ОУ-ОУ (групповая),  
CTRL\_C\_A - команда управления КС-ОС,  
CTRL\_C\_BRCST - команда управления КС (групповая),  
CTRL\_CD\_A - команда управления КС+ИС-ОС,  
CTRL\_CD\_BRCST - команда управления КС+ИС (групповая),  
CTRL\_C\_AD - команда управления КС-ОС+ИС.

Возвращаемое значение:

Двойное слово:

младшие 16 бит - первое ОС из текущей базы,

старшие 16 бит - второе ОС из текущей базы.

Выполняемые действия:

Функция позволяет получить ОС (два ОС для формата ОУ-ОУ) из текущей базы, исходя из заданного через параметр формата сообщения и КС в 0 слове текущей базы. Вызов функции имеет смысл, если из текущей базы был проведен обмен и код результата обмена подразумевает должное или возможное наличие ОС. Если ОС не было получено, возвращаемое значение неопределено. Если первого и/или второго ОС нет в указанном формате - вместо него возвращается код 0xFFFF.

### **bcgetbase**

```
unsigned bcgetbase();
```

Параметры:

Нет.

Возвращаемое значение:

Номер текущей базы.

Выполняемые действия:

Функция возвращает заданный ранее функцией bcdefbase номер базы в выбранном КК.

### **bcgetblk**

```
void bcgetblk(unsigned bcAddr, void far *pcBuffer, unsigned cwLength);
```

Параметры:

bcAddr - начальный адрес внутри базы (0 - 63);

pcBuffer - адрес массива слов в ОЗУ ПЭВМ;

cwLength - длина массива в словах (0 - 64).

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция читает указанное число слов из текущей базы ДОЗУ выбранного КК, начиная с адреса bcAddr, в ОЗУ ПЭВМ по адресу pcBuffer. При задании недопустимой длины массива возникает ошибочная ситуация с кодом ошибки BC\_BAD\_LENGTH.

### **bcgetbus**

```
unsigned bcgetbus();
```

Параметры:

Нет.

Возвращаемое значение:

Номер текущей выбранной ЛПИ.

Выполняемые действия:

Функция возвращает номер ЛПИ, заданный ранее с помощью функции bcdefbus.

### **bcgetirqmode**

```
unsigned bcgetirqmode();
```

Параметры:

Нет.

Возвращаемое значение:

Текущее значение битов режима прерываний КК.

Выполняемые действия:

Функция возвращает значение битов режима прерываний КК, заданное ранее с помощью функции bcdefirqmode.

### **bcgetlink**

```
unsigned long bcgetlink();
```

Параметры:

Нет.

Возвращаемое значение:

Двойное слово:

младшие 16 бит - следующая база в цепочке за текущей выбранной базой,  
старшие 16 бит - расширенный код управления для следующей базы.

Выполняемые действия:

Функция позволяет получить данные цепочки для текущей выбранной базы, заданные ранее функцией `bcdeflink`.

### **bcgetmaxbase**

```
unsigned bcgetmaxbase();
```

Параметры:

Нет.

Возвращаемое значение:

Максимальный номер базы.

Выполняемые действия:

Функция возвращает максимальный допустимый номер базы ДОЗУ в выбранном КК, который может использоваться в функциях `bcdefbase`, `bcstart`, `bcstartx`, `bcdeflink`.

### **bcgetmsgtime**

```
unsigned long bcgetmsgtime();
```

Параметры:

Нет.

Возвращаемое значение:

Двойное слово (32 бита) зарегистрированного аппаратного времени завершения сообщения в текущей выбранной базе.

Выполняемые действия:

Функция позволяет прочитать значение времени (значение аппаратного таймера), зарегистрированное в момент завершения выполнения сообщения КК в текущей выбранной базе.

### **bcgetstate**

```
unsigned long bcgetstate();
```

Параметры:

Нет.

Возвращаемое значение:

Двойное слово:

младшие 16 бит - база, участвующая в текущем обмене по МК,  
старшие 16 бит - содержимое регистра базового адреса (состояние).

Выполняемые действия:

Функция позволяет узнать, в какой базе в момент вызова функции выполняется цепочка сообщений. Функция доступна на устройствах ТМКХ, ТМКХI, ТА, ТАI. При вызове функции для других устройств возникает ошибочная ситуация с кодом ошибки `BC_BAD_FUNC`.

### **bcgetw**

```
unsigned bcgetw(unsigned bcAddr);
```

Параметры:

`bcAddr` - адрес внутри базы (0 - 63).

Возвращаемое значение:

Слово из текущей базы ДОЗУ.

Выполняемые действия:

Функция читает одно слово из текущей базы ДОЗУ выбранного КК по адресу `bcAddr` и возвращает его в качестве результата.



### **bcpbputblk**

```
void bcpbputblk(unsigned bcAddr, void far *pcBuffer, unsigned cwLength);
```

Параметры:

bcAddr - начальный адрес внутри базы (0 - 63);

pcBuffer - адрес массива слов в ОЗУ ПЭВМ;

cwLength - длина массива в словах (0 - 64).

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция записывает указанное число слов в текущую базу ДОЗУ выбранного КК, начиная с адреса bcAddr, из ОЗУ ПЭВМ с адреса pcBuffer. При задании недопустимой длины массива возникает ошибочная ситуация с кодом ошибки BC\_BAD\_LENGTH.

### **bcpbputw**

```
void bcpbputw(unsigned bcAddr, unsigned bcData);
```

Параметры:

bcAddr - адрес внутри базы (0 - 63);

bcData - записываемое слово.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция записывает слово bcData в текущую базу ДОЗУ выбранного КК по адресу bcAddr.

### **bcreset**

```
int bcreset();
```

Параметры:

Нет.

Возвращаемое значение:

0 - функция выполнена успешно,

TMK\_BAD\_FUNC - устройство не поддерживает работу в режиме КК.

Выполняемые действия:

Функция производит инициализацию выбранного устройства с переводом его в режим КК.

При этом на устройствах с резервированием выбирается основная ЛПИ. Значение текущей базы после вызова bcreset неопределено.

### **bcrestore [DOS]**

```
void bcrestore();
```

Параметры:

Нет.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция должна применяться только в конце пользовательских функций обработки завершения обмена при необходимости восстановления сохраненного при прерывании состояния драйвера и прервавшего КК. Функция восстанавливает сохраненный номер выбранного устройства, который был установлен в момент прерывания, а также сохраненный номер базы КК, вызвавшего прерывание.

### **bcstart**

```
int bcstart(unsigned bcBase, unsigned bcCtrlCode);
```

Параметры:

bcBase - база, из которой производится старт обмена;

bcCtrlCode - код управления, задающий формат обмена:

DATA\_BC\_RT - передача данных КК-ОУ,

DATA\_BC\_RT\_BRCST - передача данных КК-ОУ (групповая),  
DATA\_RT\_BC - передача данных ОУ-КК,  
DATA\_RT\_RT - передача данных ОУ-ОУ,  
DATA\_RT\_RT\_BRCST - передача данных ОУ-ОУ (групповая),  
CTRL\_C\_A - команда управления КС-ОС,  
CTRL\_C\_BRCST - команда управления КС (групповая),  
CTRL\_CD\_A - команда управления КС+ИС-ОС,  
CTRL\_CD\_BRCST - команда управления КС+ИС (групповая),  
CTRL\_C\_AD - команда управления КС-ОС+ИС.

Возвращаемое значение:

0 - функция выполнена успешно;

BC\_BAD\_BASE - заданное значение базы находится вне допустимого диапазона.

Выполняемые действия:

Функция иницирует начало обмена по ЛПИ МК, заданной заранее в вызове bcdefbus, из базы bcBase ДОЗУ выбранного КК в формате, заданном bcCtrlCode и командными словами сообщения, сформированного предварительно в этой базе. Вызов функции с недопустимым значением базы приводит к ошибочной ситуации с кодом ошибки BC\_BAD\_BASE. Эта функция работает на всех устройствах и позволяет запустить только одиночный обмен из указанной базы. По завершению обмена формируется обычный код завершения обмена и вызывается пользовательская функция обработки завершения обмена заданная через bcdefintnorm или bcdefinexs.

### **bcstartx**

```
int bcstartx(unsigned bcBase, unsigned bcCtrlCodeX);
```

Параметры:

bcBase - база, из которой производится старт обмена;

bcCtrlCodeX - расширенный код управления, объединяющий по "ИЛИ" формат обмена:

DATA\_BC\_RT - передача данных КК-ОУ,  
DATA\_BC\_RT\_BRCST - передача данных КК-ОУ (групповая),  
DATA\_RT\_BC - передача данных ОУ-КК,  
DATA\_RT\_RT - передача данных ОУ-ОУ,  
DATA\_RT\_RT\_BRCST - передача данных ОУ-ОУ (групповая),  
CTRL\_C\_A - команда управления КС-ОС,  
CTRL\_C\_BRCST - команда управления КС (групповая),  
CTRL\_CD\_A - команда управления КС+ИС-ОС,  
CTRL\_CD\_BRCST - команда управления КС+ИС (групповая),  
CTRL\_C\_AD - команда управления КС-ОС+ИС;

признак продолжения:

CX\_CONT - по окончании обмена перейти в следующую базу цепочки, если не было ошибки или установленных битов ОС,

CX\_STOP - остановиться по окончании обмена;

номер ЛПИ:

CX\_BUS\_A (или CX\_BUS\_0) - основная ЛПИ,

CX\_BUS\_B (или CX\_BUS\_1) - резервная ЛПИ;

требование формирования сигнального прерывания:

CX\_SIG - сформировать сигнальное прерывание при старте обмена,

CX\_NOSIG - не формировать сигнальное прерывание.

Возвращаемое значение:

0 - функция выполнена успешно;

BC\_BAD\_BASE - заданное значение базы находится вне допустимого диапазона;

BC\_BAD\_FUNC - выбранный КК не поддерживает старт обмена через bcstartx.

Выполняемые действия:

Функция иницирует начало обмена по ЛПИ МК, заданной bcCtrlCodeX, из базы bcBase ДОЗУ выбранного КК в формате, заданном bcCtrlCodeX и командными словами

сообщения, сформированного предварительно в этой базе. Вызов функции с недопустимым значением базы приводит к ошибочной ситуации с кодом ошибки BC\_BAD\_BASE. Эта функция работает на устройствах ТМКХ, ТМКХІ, ТА, ТАІ. Она позволяет запустить как одиночный обмен из указанной базы, так и цепочку обменов, начиная с заданной базы. Старт цепочки задается включением CX\_CONT в bcCtrlCodeX. Цепочка должна быть заранее сформирована через вызовы bcdeflink. По завершению обмена формируется расширенный код завершения обмена и вызывается пользовательская функция обработки завершения обмена заданная через bcdefintx. Если в процессе выполнения цепочки в управляющих словах будут встречаться установленные биты сигнальных прерываний, то при старте этих сообщений цепочки будет вызываться пользовательская функция обработки сигнальных прерываний, заданная через bcdefintsig.

### **bcstop**

```
int bcstop();
```

Параметры:

Нет.

Возвращаемое значение:

0 - функция выполнена успешно;

BC\_BAD\_FUNC - выбранный КК не поддерживает старт обмена через bcstartx и останов по bcstop.

Выполняемые действия:

Функция обнуляет внутренний признак продолжения цепочки текущего выбранного КК, что приводит к останову цепочки после завершения текущего обмена цепочки, запущенной вызовом bcstartx. Функция работает на устройствах ТМКХ, ТМКХІ, ТА, ТАІ.

\*\*\*\*\*

### **Функции для Монитора (устройства ТМКХ, ТМКХІ, ТА, ТАІ)**

\*\*\*\*\*

### **mtdefbase**

```
int mtdefbase(unsigned mtBasePC);
```

Параметры:

mtBasePC - устанавливаемая база.

Возвращаемое значение:

0 - функция выполнена успешно;

MT\_BAD\_BASE - заданное значение базы находится вне допустимого диапазона.

Выполняемые действия:

Функция настраивает выбранный МТ и драйвер на дальнейшую работу с ДОЗУ в указанной базе. Вызов функции с недопустимым значением базы приводит к ошибочной ситуации с кодом ошибки MT\_BAD\_BASE.

### **mtdefintsig [DOS]**

```
void mtdefintsig(void (far* UserSigMT)(unsigned));
```

Параметры:

UserSigMT - пользовательская функция, вызываемая при возникновении сигнального прерывания во время работы МТ, запущенного функцией mtstartx (прерывание mtIntSig); функция должна быть описана как far-функция, возвращающая тип void, и получающая через параметры одно слово: номер базы, в которой возникло сигнальное прерывание.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция задает драйверу имя пользовательской функции, которая будет вызываться каждый раз при возникновении сигнального прерывания во время работы МТ,

запущенного функцией `mtstartx`. Если время выполнения пользовательской функции обработки сигнальных прерываний больше периода их возникновения, возможен пропуск вызовов пользовательской функции для промежуточных прерываний. Пользовательская функция должна сама определять такую ситуацию по получаемому параметру (номеру базы возникновения прерывания).

### **mtdefintx [DOS]**

```
void mtdefintx(void (far* UserXMT)(unsigned, unsigned));
```

Параметры:

UserXMT - пользовательская функция, вызываемая при завершении работы МТ, запущенного функцией `mtstartx` (прерывание `mtIntX`); функция должна быть описана как `far`-функция, возвращающая тип `void`, и получающая через параметры два слова: номер базы, в которой произошел останов, слово расширенного результата обмена.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция задает драйверу имя пользовательской функции, которая будет вызываться каждый раз при завершении работы выбранным МТ, если запущенной функцией `mtstartx`.

### **mtdefirqmode**

```
int mtdefirqmode(unsigned mtIrqMode);
```

Параметры:

`mtIrqMode` - битовая маска управления режимом прерываний МТ:

`TMK_IRQ_OFF` - задание этого бита отключает прерывание устройств `TMKX`, `TA` от шины `ISA`;

`MT_GENER1_BL` - задание этого бита блокирует прерывания по обнаружению генерации в основной ЛПИ устройства `TMKX`;

`MT_GENER2_BL` - задание этого бита блокирует прерывания по обнаружению генерации в резервной ЛПИ устройства `TMKX`.

Возвращаемое значение:

0 - функция выполнена успешно;

`MT_BAD_FUNC` - выбранный МТ не поддерживает программное управление режимом прерываний МТ.

Выполняемые действия:

Функция задает режим прерываний выбранного МТ. По умолчанию всегда устанавливаются биты `MT_GENER1_BL` и `MT_GENER2_BL`, так как признак генерации может возникать асинхронно, а никакой обработки признака в драйвере нет.

Единственной правильной реакцией на возникновение признака генерации может быть сброс устройства вызовом `mtreset` или блокировка признаков генерации в пользовательской функции обработки прерывания. Если биты признаков генерации не анализируются в пользовательских функциях обработки прерываний, запрещается снимать блокировки прерываний по генерации.

### **mtdeflink**

```
int mtdeflink(unsigned mtBase, unsigned mtCtrlCodeX);
```

Параметры:

`mtBase` - база, следующая в цепочке за текущей выбранной `mtdefbase`;

`mtCtrlCodeX` - расширенный код управления для базы `mtBase`, объединяющий по "ИЛИ": признак продолжения:

`CX_CONT` - по окончании обмена перейти в следующую базу цепочки,

`CX_STOP` - остановиться по окончании обмена;

признак останова (прерывания) по ошибке:

`CX_INT` - останов МТ при обнаружении ошибки или установленного бита ОС,

`CX_NOINT` - нет останова МТ при обнаружении ошибки или установленного бита ОС;

требование формирования сигнального прерывания:

CX\_SIG - сформировать сигнальное прерывание при старте обмена,

CX\_NOSIG - не формировать сигнальное прерывание.

Возвращаемое значение:

0 - функция выполнена успешно;

MT\_BAD\_BASE - заданное значение базы находится вне допустимого диапазона;

MT\_BAD\_FUNC - выбранный MT не поддерживает старт обмена через mtstartx и формирование цепочек.

Выполняемые действия:

Функция добавляет в цепочку после текущей выбранной базы базу mtBase выбранного MT. Если затем при старте mtstartx управление по цепочке дойдет до текущей базы и в расширенном коде управления для текущей базы будет стоять признак продолжения CX\_CONT, то следующим обменом в цепочке будет обмен из базы mtBase с кодом управления mtCtrlCodeX. Цепочка остановится, если выполнится обмен с признаком CX\_STOP в коде управления, также цепочка остановится по любой ошибке или установленному биту в принятом ОС, если будет установлен признак CX\_INT. Вызов функции с недопустимым значением базы приводит к ошибочной ситуации с кодом ошибки MT\_BAD\_BASE.

### **mtgetsw**

```
unsigned mtgetsw();
```

Параметры:

Нет.

Возвращаемое значение:

Слово расширенного результата обмена для текущей базы.

Выполняемые действия:

Функция позволяет получить слово расширенного результата обмена из текущей базы для тех случаев, когда монитор записал сообщение в базу и не остановился, а перешел в следующую в цепочке базу. При останове монитора слово расширенного результата обмена для последней базы можно получить только из параметров функции, вызываемой по прерыванию останова MT, заданной в mtdefintx.

### **mtgetbase**

```
unsigned mtgetbase();
```

Параметры:

Нет.

Возвращаемое значение:

Номер текущей базы.

Выполняемые действия:

Функция возвращает заданный ранее функцией mtdefbase номер базы в выбранном MT.

### **mtgetblk**

```
void mtgetblk(unsigned mtAddr, void far *pcBuffer, unsigned cwLength);
```

Параметры:

mtAddr - начальный адрес внутри базы (0 - 63);

pcBuffer - адрес массива слов в ОЗУ ПЭВМ;

cwLength - длина массива в словах (0 - 64).

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция читает указанное число слов из текущей базы ДОЗУ выбранного MT, начиная с адреса mtAddr, в ОЗУ ПЭВМ по адресу pcBuffer. При задании недопустимой длины массива возникает ошибочная ситуация с кодом ошибки MT\_BAD\_LENGTH.

### **mtgetirqmode**

`unsigned mtgetirqmode();`

Параметры:

Нет.

Возвращаемое значение:

Текущее значение битов режима прерываний МТ.

Выполняемые действия:

Функция возвращает значение битов режима прерываний МТ, заданное ранее с помощью функции `mtdefirqmode`.

### **mtgetlink**

`unsigned long mtgetlink();`

Параметры:

Нет.

Возвращаемое значение:

Двойное слово:

младшие 16 бит - следующая база в цепочке за текущей выбранной базой,

старшие 16 бит - расширенный код управления для следующей базы.

Выполняемые действия:

Функция позволяет получить данные цепочки для текущей выбранной базы, заданные ранее функцией `mtdeflink`.

### **mtgetmaxbase**

`unsigned mtgetmaxbase();`

Параметры:

Нет.

Возвращаемое значение:

Максимальный номер базы.

Выполняемые действия:

Функция возвращает максимальный допустимый номер базы ДОЗУ в выбранном МТ, который может использоваться в функциях `mtdefbase`, `mtstartx`, `mtdeflink`.

### **mtgetmsgtime**

`unsigned long mtgetmsgtime();`

Параметры:

Нет.

Возвращаемое значение:

Двойное слово (32 бита) зарегистрированного аппаратного времени завершения сообщения в текущей выбранной базе.

Выполняемые действия:

Функция позволяет прочитать значение времени (значение аппаратного таймера), зарегистрированное в момент завершения сообщения МТ в текущей выбранной базе.

### **mtgetstate**

`unsigned long mtgetstate();`

Параметры:

Нет.

Возвращаемое значение:

Двойное слово:

младшие 16 бит - база, участвующая в текущем обмене по МК,

старшие 16 бит - содержимое регистра базового адреса (состояние).

Выполняемые действия:

Функция позволяет узнать, в какой базе в момент вызова функции выполняется цепочка сообщений монитора.

### **mtgetw**

```
unsigned mtgetw(unsigned mtAddr);
```

Параметры:

mtAddr - адрес внутри базы (0 - 63).

Возвращаемое значение:

Слово из текущей базы ДОЗУ.

Выполняемые действия:

Функция читает одно слово из текущей базы ДОЗУ выбранного МТ по адресу mtAddr и возвращает его в качестве результата.

### **mtputblk**

```
void mtputblk(unsigned mtAddr, void far *pcBuffer, unsigned cwLength);
```

Параметры:

mtAddr - начальный адрес внутри базы (0 - 63);

pcBuffer - адрес массива слов в ОЗУ ПЭВМ;

cwLength - длина массива в словах (0 - 64).

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция записывает указанное число слов в текущую базу ДОЗУ выбранного МТ, начиная с адреса mtAddr, из ОЗУ ПЭВМ с адреса pcBuffer. При задании недопустимой длины массива возникает ошибочная ситуация с кодом ошибки MT\_BAD\_LENGTH.

### **mtputw**

```
void mtputw(unsigned mtAddr, unsigned mtData);
```

Параметры:

mtAddr - адрес внутри базы (0 - 63);

mtData - записываемое слово.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция записывает слово mtData в текущую базу ДОЗУ выбранного МТ по адресу mtAddr.

### **mtreset**

```
int mtreset();
```

Параметры:

Нет.

Возвращаемое значение:

0 - функция выполнена успешно,

TMK\_BAD\_FUNC - устройство не поддерживает работу в режиме МТ.

Выполняемые действия:

Функция производит инициализацию выбранного устройства с переводом его в режим МТ. Значение текущей базы после вызова mtreset неопределено.

### **mtrestore [DOS]**

```
void mtrestore();
```

Параметры:

Нет.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция должна применяться только в конце пользовательских функций обработки завершения обмена при необходимости восстановления сохраненного при прерывании состояния драйвера и прервавшего МТ. Функция восстанавливает сохраненный номер выбранного устройства, который был установлен в момент прерывания, а также сохраненный номер базы МТ, вызвавшего прерывание.

### **mtstartx**

```
int mtstartx(unsigned mtBase, unsigned mtCtrlCodeX);
```

Параметры:

mtBase - база, из которой производится старт обмена;

mtCtrlCodeX - расширенный код управления, объединяющий по "ИЛИ" признак продолжения:

CX\_CONT - по окончании обмена перейти в следующую базу цепочки,

CX\_STOP - остановиться по окончании обмена;

признак останова (прерывания) по ошибке:

CX\_INT - останов МТ при обнаружении ошибки или установленного бита ОС,

CX\_NOINT - нет останова МТ при обнаружении ошибки или установленного бита ОС;

требование формирования сигнального прерывания:

CX\_SIG - сформировать сигнальное прерывание при старте обмена,

CX\_NOSIG - не формировать сигнальное прерывание.

Возвращаемое значение:

0 - функция выполнена успешно;

MT\_BAD\_BASE - заданное значение базы находится вне допустимого диапазона;

MT\_BAD\_FUNC - выбранный МТ не поддерживает старт обмена через mtstartx.

Выполняемые действия:

Функция иницирует начало работы МК из базы mtBase ДОЗУ выбранного МТ в формате, с признаками для первой базы, заданными mtCtrlCodeX. Вызов функции с недопустимым значением базы приводит к ошибочной ситуации с кодом ошибки MT\_BAD\_BASE. Эта функция позволяет запустить как ожидание одного сообщения, так и ожидание нескольких сообщений в базы сформированной цепочки, начиная с заданной базы. Старт цепочки задается включением CX\_CONT в mtCtrlCodeX. Цепочка должна быть заранее сформирована через вызовы mtdeflink. По завершению обмена формируется расширенный код завершения обмена и вызывается пользовательская функция обработки завершения обмена заданная через mtdefintx. Если в процессе выполнения цепочки в управляющих словах будут встречаться установленные биты сигнальных прерываний, то при старте этих сообщений цепочки будет вызываться пользовательская функция обработки сигнальных прерываний, заданная через mtdefintsig.

### **mtstop**

```
int mtstop();
```

Параметры:

Нет.

Возвращаемое значение:

0 - функция выполнена успешно;

MT\_BAD\_FUNC - выбранный МТ не поддерживает старт обмена через mtstartx и останов по mtstop.

Выполняемые действия:

Функция обнуляет внутренний признак продолжения цепочки текущего выбранного МТ, что приводит к останову цепочки после завершения приема текущего сообщения цепочки, запущенной вызовом mtstartx. Смысл использования этой функции есть только, если известно, что сообщения идут в МК непрерывно. Если в канале сообщений нет, то и нет смысла в вызове mtstop, так как монитор все равно будет ждать хотя бы одного сообщения, после которого остановится. В общем случае остановить работу монитора



можно вызовом `mtreset`. При этом МТ сразу сбросится, оборвав прием сообщения, даже если он его вел, и не выдаст прерываний по окончанию цепочки.

\*\*\*\*\*  
**Функции для Оконечного Устройства**  
\*\*\*\*\*

### **rtbusy**

`unsigned rtbusy();`

Параметры:

Нет.

Возвращаемое значение:

0 - выбранный подадрес свободен;

1 - выбранный подадрес занят обменом с МК.

Выполняемые действия:

Функция определяет факт занятости текущего подадреса выбранного ОУ в режиме работы без флагов.

### **rtclranswbits**

`void rtclranswbits(unsigned rtClrBits);`

Параметры:

`rtClrBits` - маска, задающая сброс в 0 указанных бит ОС:

SREQ - бит "Запрос обслуживания подсистемы",

BUSY - бит "Подсистема занята",

SSFL - бит "Неисправность подсистемы",

RTFL - бит "Неисправность терминала",

DNBA - бит "Готов принять управление каналом".

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция очищает заданные биты ответных слов, выдаваемых выбранным ОУ.

### **rtclrflag**

`void rtclrflag();`

Параметры:

Нет.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция сбрасывает флаг в флаговом слове текущего подадреса выбранного ОУ в режиме работы ОУ с флагами.

### **rtdefaddress**

`int rtdefaddress(unsigned rtAddress);`

Параметры:

`rtAddress` - адрес ОУ в МК (0 - 30).

Возвращаемое значение:

0 - функция выполнена успешно;

RT\_BAD\_ADDRESS - задан недопустимый адрес ОУ;

RT\_BAD\_FUNC - выбранное ОУ не поддерживает программное задание адреса ОУ в МК.

Выполняемые действия:

Функция программирует адрес выбранного ОУ в МК. Если тип устройства выбранного ОУ не поддерживает программирование адреса (адрес устанавливается перемычками на устройстве), возникает ошибочная ситуация с кодом ошибки RT\_BAD\_FUNC. При задании

недопустимого адреса ОУ возникает ошибочная ситуация с кодом ошибки RT\_BAD\_ADDRESS.

### **rtdefintcmd [DOS]**

```
void rtdefintcmd(void (far* UserCmdRT)(unsigned));
```

Параметры:

UserCmdRT - пользовательская функция, вызываемая при поступлении команды режима управления (прерывание rtIntCmd); функция должна быть описана как far-функция, возвращающая тип void, и получающая через параметр одно слово: код поступившей команды, включающий саму команду (младшие 5 бит) и бит "прием/передача" командного слова данной команды.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция задает драйверу имя пользовательской функции, которая будет вызываться каждый раз при поступлении команды режима управления в выбранное ОУ, которая не может быть выполнена аппаратно самим устройством (см. руководство программиста).

### **rtdefintdata [DOS]**

```
void rtdefintdata(void (far* UserDataRT)(unsigned));
```

Параметры:

UserDataRT - пользовательская функция, вызываемая после выполнения ОУ команды приема/передачи данных (прерывание rtIntData); функция должна быть описана как far-функция, возвращающая тип void, и получающая через параметр одно слово: слово состояния ОУ, содержащее младшие 11 битов КС.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция задает драйверу имя пользовательской функции, которая будет вызываться каждый раз после выполнения ОУ команды приема/передачи данных. Не все устройства способны аппаратно формировать такое прерывание, поэтому реально заданная пользовательская функция будет вызываться только на устройствах ТМКХ, ТМКХI, ТА, ТАI. По умолчанию формирование этих прерываний заблокировано, поэтому, чтобы получать их, необходимо их разблокировать вызовом rtdefirqmode(rtgetirqmode())&~RT\_DATA\_BL).

### **rtdefinterr [DOS]**

```
void rtdefinterr(void (far* UserErrRT)(unsigned));
```

Параметры:

UserErrRT - пользовательская функция, вызываемая при обнаружении ошибки МК при обмене выбранного ОУ с каналом (прерывание rtIntErr); функция должна быть описана как far-функция, возвращающая тип void, и получающая через параметр одно слово: слово состояния ОУ с установленным битом "Ошибка сообщения МК" и битами командного слова, при обработке которого возникла ошибка.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция задает драйверу имя пользовательской функции, которая будет вызываться каждый раз при возникновении ошибки обмена с МК в выбранном ОУ (см. руководство программиста).

### **rtdefirqmode**

```
int rtdefirqmode(unsigned rtIrqMode);
```

Параметры:

rtIrqMode - битовая маска управления режимом прерываний ОУ:

TMK\_IRQ\_OFF - задание этого бита отключает прерывание устройств TMKX, ТА от шины ISA;

RT\_GENER1\_BL - задание этого бита блокирует прерывания по обнаружению генерации в основной ЛПИ устройства TMKX;

RT\_GENER2\_BL - задание этого бита блокирует прерывания по обнаружению генерации в резервной ЛПИ устройства TMKX;

RT\_DATA\_BL - задание этого бита блокирует прерывания по командам приема/передачи данных.

Возвращаемое значение:

0 - функция выполнена успешно;

RT\_BAD\_FUNC - выбранное ОУ не поддерживает программное управление режимом прерываний ОУ.

Выполняемые действия:

Функция задает режим прерываний выбранного ОУ. По умолчанию всегда устанавливаются биты RT\_GENER1\_BL и RT\_GENER2\_BL, так как признак генерации может возникать асинхронно, а никакой обработки признака в драйвере нет.

Единственной правильной реакцией на возникновение признака генерации может быть сброс устройства вызовом rreset или блокировка признаков генерации в пользовательской функции обработки прерывания. Если биты признаков генерации не анализируются в пользовательских функциях обработки прерываний, запрещается снимать блокировки прерываний по генерации. Также по умолчанию установлен бит RT\_DATA\_BL, поэтому ОУ не выдает прерываний по командам приема/передачи данных. Чтобы разблокировать эти прерывания можно вызвать rtdelfirqmode(rtgetirqmode()&~RT\_DATA\_BL).

## **rtgetirqmode**

```
unsigned rtgetirqmode();
```

Параметры:

Нет.

Возвращаемое значение:

Текущее значение битов режима прерываний ОУ.

Выполняемые действия:

Функция возвращает значение битов режима прерываний ОУ, заданное ранее с помощью функции rtdelfirqmode.

## **rtdefmode**

```
int rtdefmode(unsigned rtMode);
```

Параметры:

rtMode - режим работы ОУ, задаваемый комбинацией (объединением по ИЛИ) следующих битов режима:

RT\_HBIT\_MODE - включает режим работы ОУ с использованием аппаратного бита в поле подадреса командного слова (по умолчанию режим включен);

RT\_FLAG\_MODE - включает режим работы ОУ с флагами (по умолчанию режим выключен);

RT\_BRCDST\_MODE - включает режим работы ОУ, при котором воспринимаются командные слова с групповым адресом (по умолчанию режим включен).

Возвращаемое значение:

0 - функция выполнена успешно;

RT\_BAD\_FUNC - выбранное ОУ не поддерживает программное задание режимов работы ОУ.

Выполняемые действия:

Функция программирует режимы работы выбранного ОУ; функция rreset, вызванная непосредственно после вызова tmkconfig, устанавливает ОУ в режим (RT\_HBIT\_MODE |

RT\_BRCST\_MODE), следующие вызовы rreset не изменяют текущих режимов ОУ; устройства TMK400, TMKMPC не допускают программного управления режимами ОУ, на этих устройствах режимы ОУ задаются установкой переключателей на устройстве, а вызов функции для этих типов устройств вызывает ошибочную ситуацию с кодом ошибки RT\_BAD\_FUNC; полное управление всеми режимами допускают устройства RTMK400, TA, TAI; устройства TMKX, TMKXI допускают управление RT\_FLAG\_MODE и RT\_BRCST\_MODE.

### **rtdefpage**

```
int rtdefpage(unsigned rtPage);
```

Параметры:

rtPage - выбираемая страница ДОЗУ.

Возвращаемое значение:

0 - функция выполнена успешно;

RT\_BAD\_PAGE - задан недопустимый номер страницы ДОЗУ.

Выполняемые действия:

Функция устанавливает в выбранном ОУ страницу rtPage в качестве текущей. При вызове с недопустимым для выбранного ОУ значением номера страницы возникает ошибочная ситуация с кодом ошибки RT\_BAD\_PAGE.

### **rtdefpagebus**

```
int rtdefpagebus(unsigned rtPageBus);
```

Параметры:

rtPageBus - выбираемая страница ДОЗУ для доступа со стороны МК.

Возвращаемое значение:

0 - функция выполнена успешно;

RT\_BAD\_PAGE - задан недопустимый номер страницы ДОЗУ;

RT\_BAD\_FUNC - выбранное ОУ не поддерживает раздельного задания страницы для МК и ПЭВМ.

Выполняемые действия:

Функция устанавливает в выбранном ОУ страницу rtPageBus доступной для МК, не влияя на страницу, доступную ПЭВМ. При вызове с недопустимым для выбранного ОУ значением номера страницы возникает ошибочная ситуация с кодом ошибки RT\_BAD\_PAGE. Если тип устройства выбранного ОУ не поддерживает раздельное задание страниц для МК и ПЭВМ, возникает ошибочная ситуация с кодом ошибки RT\_BAD\_FUNC.

### **rtdefpagepc**

```
int rtdefpagepc(unsigned rtPagePC);
```

Параметры:

rtPagePC - выбираемая страница ДОЗУ для доступа со стороны ПЭВМ.

Возвращаемое значение:

0 - функция выполнена успешно;

RT\_BAD\_PAGE - задан недопустимый номер страницы ДОЗУ;

RT\_BAD\_FUNC - выбранное ОУ не поддерживает раздельного задания страницы для МК и ПЭВМ.

Выполняемые действия:

Функция устанавливает в выбранном ОУ страницу rtPagePC в качестве текущей доступной для ПЭВМ, не влияя на страницу, доступную МК. При вызове с недопустимым для выбранного ОУ значением номера страницы возникает ошибочная ситуация с кодом ошибки RT\_BAD\_PAGE. Если тип устройства выбранного ОУ не поддерживает раздельное задание страниц для МК и ПЭВМ, возникает ошибочная ситуация с кодом ошибки RT\_BAD\_FUNC.

### **rtdefsubaddr**

```
void rtdefsubaddr(unsigned rtDir, unsigned rtSubAddr);
```

Параметры:

rtDir - выбор подадреса приема или подадреса передачи:

RT\_RECEIVE - подадрес приема,

RT\_TRANSMIT - подадрес передачи;

rtSubAddr - номер подадреса:

1 - 30 в режиме без аппаратного бита,

16 - 30 в режиме с аппаратным битом.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция настраивает выбранное ОУ и драйвер на дальнейшую работу с ДОЗУ в указанном подадресе.

### **rtenable**

```
unsigned rtenable(unsigned rtEnable);
```

Параметры:

rtEnable - задание операции функции:

RT\_ENABLE - включение (разрешение работы) ОУ,

RT\_DISABLE - выключение (запрет работы) ОУ,

RT\_GET\_ENABLE - чтение текущего значения разрешения работы ОУ;

Возвращаемое значение (при rtEnable != RT\_GET\_ENABLE):

0 - функция выполнена успешно;

RT\_BAD\_FUNC - ошибка задания параметра.

Возвращаемое значение (при rtEnable == RT\_GET\_ENABLE):

RT\_ENABLE - ОУ включено,

RT\_DISABLE - ОУ выключено.

Выполняемые действия:

Функция позволяет включать/выключать выбранное ОУ, не выключая сам режим ОУ.

Основное назначение функции - использование на устройствах, работающих в режиме многоадресного ОУ, т.е. управление виртуальными ОУ в рамках одного физического устройства. Но, кроме того, также возможно выполнение этой функции на устройствах типов RTMK400, TMKX, TMKXI, TA, TAI в режиме ОУ. Устройства типов TMK400 и TMKMPC не содержат аппаратных средств, необходимых для правильной работы этой функции.

### **rtgetaddress**

```
unsigned rtgetaddress();
```

Параметры:

Нет.

Возвращаемое значение:

Адрес ОУ в МК.

Выполняемые действия:

Функция читает адрес выбранного ОУ в МК из самого ОУ (если позволяет тип применяемого устройства) или из внутренней переменной драйвера. Если тип устройства выбранного ОУ не поддерживает программирование адреса (адрес устанавливается переключками на устройстве) и не поддерживает программное определение установленного адреса, возникает ошибочная ситуация с кодом ошибки RT\_BAD\_FUNC.

### **rtgetanswbits**

```
unsigned rtgetanswbits();
```

Параметры:

Нет.

Возвращаемое значение:

Состояние битов поля флагов в ОС:

SREQ - бит "Запрос обслуживания подсистемы",

BUSY - бит "Подсистема занята",

SSFL - бит "Неисправность подсистемы",

RTFL - бит "Неисправность терминала",

DNBA - бит "Готов принять управление каналом".

Выполняемые действия:

Функция возвращает текущее состояние битов, определяющих значение соответствующих битов поля флагов ОС выбранного ОУ, заданных ранее функциями rtsetanswbits и rtclranswbits.

### **rtgetblk**

```
void rtgetblk(unsigned rtAddr, void far *pcBuffer, unsigned cwLength);
```

Параметры:

rtAddr - начальный адрес внутри подадреса (0 - 31);

pcBuffer - адрес массива слов в ОЗУ ПЭВМ;

cwLength - длина массива в словах (0 - 32).

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция читает указанное число слов из текущего подадреса ДОЗУ выбранного ОУ, начиная с адреса rtAddr, в ОЗУ ПЭВМ по адресу pcBuffer. При задании недопустимой длины массива возникает ошибочная ситуация с кодом ошибки RT\_BAD\_LENGTH.

### **rtgetcddata**

```
unsigned rtgetcddata(unsigned rtBusCommand);
```

Параметры:

rtBusCommand - команда режима управления с присоединенным ИС.

Возвращаемое значение:

Присоединенное ИС для указанной команды.

Выполняемые действия:

Функция читает присоединенное ИС для заданной команды режима управления (должен быть задан код команды и бит "прием/передача"). ВНИМАНИЕ! После вызова этой функции значение текущего подадреса не определено.

### **rtgetflag**

```
unsigned rtgetflag(unsigned rtDir, unsigned rtSubAddr);
```

Параметры:

rtDir - выбор подадреса приема или подадреса передачи:

RT\_RECEIVE - подадрес приема,

RT\_TRANSMIT - подадрес передачи;

rtSubAddr - номер подадреса:

1 - 30 в режиме без аппаратного бита,

16 - 30 в режиме с аппаратным битом.

Возвращаемое значение:

Флаговое слово для указанного подадреса.

Выполняемые действия:

Функция читает флаговое слово для заданного подадреса в выбранном ОУ и настраивает выбранное ОУ и драйвер на дальнейшую работу с ДОЗУ в этом подадресе.

### **rtgetflags**

```
void rtgetflags(void far *pcBuffer, unsigned rtDir, unsigned
rtFlagMin, unsigned rtFlagMax);
```

Параметры:

pcBuffer - адрес массива для чтения блока флаговых слов в ОЗУ ПЭВМ;

rtDir - выбор подадресов приема или подадресов передачи:

RT\_RECEIVE - подадреса приема,

RT\_TRANSMIT - подадреса передачи;

rtFlagMin, rtFlagMax - диапазон подадресов, для которых должны читаться флаговые слова (1 - 30).

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция читает из ДОЗУ блок флаговых слов для заданного диапазона подадресов выбранного ОУ. Если rtFlagMin==rtFlagMax, читается одно флаговое слово. Если rtFlagMin>rtFlagMax, функция не выполняет никакого чтения. ВНИМАНИЕ! После вызова этой функции значение текущего подадреса не определено.

### **rtgetmaxpage**

```
unsigned rtgetmaxpage();
```

Параметры:

Нет.

Возвращаемое значение:

Максимальный номер страницы.

Выполняемые действия:

Функция возвращает максимальный допустимый номер страницы ДОЗУ в выбранном ОУ, который может использоваться в функциях rtdefpage, rtdefpagebus и rtdefpagepc.

### **rtgetmode**

```
unsigned rtgetmode();
```

Параметры:

Нет.

Возвращаемое значение:

Текущий режим работы ОУ, индицируемый комбинацией установленных битов режима:

RT\_HBIT\_MODE - включен режим работы ОУ с использованием аппаратного бита в поле подадреса командного слова;

RT\_FLAG\_MODE - включен режим работы ОУ с флагами;

RT\_BRCST\_MODE - включен режим работы ОУ, при котором воспринимаются командные слова с групповым адресом;

Для типов устройств, которые не поддерживают программное задание режимов работы ОУ, всегда возвращается 0.

Выполняемые действия:

Функция возвращает текущие режимы работы выбранного ОУ для устройств RTMK400, TMKX, TMKXI, TA, TAI; функция rreset, вызванная непосредственно после вызова tmkconfig, устанавливает ОУ в режим (RT\_HBIT\_MODE | RT\_BRCST\_MODE), следующие вызовы rreset не изменяют текущих режимов ОУ; устройства TMKX, TMKXI не имеют программного управления режимом RT\_HBIT\_MODE, который задается установкой перемычки на устройстве; устройства TMK400, TMKMPC не допускают программного управления режимами ОУ, на этих устройствах режимы ОУ задаются установкой перемычек на устройстве, а вызов функции для этих типов устройств вызывает ошибочную ситуацию с кодом ошибки RT\_BAD\_FUNC.

### **rtgetmsgtime**

```
unsigned long rtgetmsgtime();
```

Параметры:

Нет.

Возвращаемое значение:

Двойное слово (32 бита) зарегистрированного аппаратного времени завершения сообщения в текущем выбранном подадресе.

Выполняемые действия:

Функция позволяет прочитать значение времени (значение аппаратного таймера), зарегистрированное в момент завершения выполнения сообщения ОУ в текущем выбранном подадресе.

### **rtgetpage**

```
unsigned rtgetpage();
```

Параметры:

Нет.

Возвращаемое значение:

Номер текущей страницы.

Выполняемые действия:

Функция возвращает заданный ранее функцией rtdefpage номер страницы в выбранном ОУ.

### **rtgetpagebus**

```
unsigned rtgetpagebus();
```

Параметры:

Нет.

Возвращаемое значение:

Номер текущей страницы, доступной со стороны МК.

Выполняемые действия:

Функция возвращает заданный ранее функцией rtdefpagebus номер страницы в выбранном ОУ, доступной со стороны МК.

### **rtgetpagepc**

```
unsigned rtgetpagepc();
```

Параметры:

Нет.

Возвращаемое значение:

Номер текущей страницы, доступной со стороны ПЭВМ.

Выполняемые действия:

Функция возвращает заданный ранее функцией rtdefpagepc номер страницы в выбранном ОУ, доступной со стороны ПЭВМ.

### **rtgetstate**

```
unsigned rtgetstate();
```

Параметры:

Нет.

Возвращаемое значение:

Слово состояния ОУ.

Выполняемые действия:

Функция читает слово состояния выбранного ОУ, содержащее биты последнего КС, принятого ОУ, бит занятости подадреса, бит ошибки сообщения МК и бит ошибки обмена с подсистемой (только для устройств ТМК400, RTМК400, ТМКМРС).

### **rtgetsubaddr**

```
unsigned rtgetsubaddr();
```

Параметры:

Нет.



Возвращаемое значение:

Номер текущего подадреса с битом "прием/передача".

Выполняемые действия:

Функция возвращает номер текущего подадреса выбранного ОУ, заданного ранее функциями `rtdefsubaddr`, `rtlock`, `rtgetflag`.

### **rtgetw**

```
unsigned rtgetw(unsigned rtAddr);
```

Параметры:

`rtAddr` - адрес внутри подадреса (0 - 31).

Возвращаемое значение:

Слово из текущего подадреса ДОЗУ.

Выполняемые действия:

Функция читает одно слово из текущего подадреса ДОЗУ выбранного ОУ по адресу `rtAddr` и возвращает его в качестве результата.

### **rtlock**

```
void rtlock(unsigned rtDir, unsigned rtSubAddr);
```

Параметры:

`rtDir` - выбор подадреса приема или подадреса передачи:

`RT_RECEIVE` - подадрес приема,

`RT_TRANSMIT` - подадрес передачи;

`rtSubAddr` - номер подадреса:

1 - 30 в режиме без аппаратного бита,

16 - 30 в режиме с аппаратным битом.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция настраивает выбранное ОУ и драйвер на дальнейшую работу с ДОЗУ в указанном подадресе и при этом блокирует подадрес для доступа со стороны МК. Если в момент вызова функции шел обмен МК с этим подадресом, блокировка задерживается до окончания обмена. Успешность блокировки можно проверить последующим вызовом `rtbusy`.

### **rtputblk**

```
void rtputblk(unsigned rtAddr, void far *pcBuffer, unsigned cwLength);
```

Параметры:

`rtAddr` - начальный адрес внутри подадреса (0 - 31);

`pcBuffer` - адрес массива слов в ОЗУ ПЭВМ;

`cwLength` - длина массива в словах (0 - 32).

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция записывает указанное число слов в текущий подадрес ДОЗУ выбранного ОУ, начиная с адреса `rtAddr`, из ОЗУ ПЭВМ с адреса `pcBuffer`. При задании недопустимой длины массива возникает ошибочная ситуация с кодом ошибки `RT_BAD_LENGTH`.

### **rtputcmddata**

```
void rtputcmddata(unsigned rtBusCommand, unsigned rtData);
```

Параметры:

`rtBusCommand` - команда режима управления с присоединенным ИС;

`rtData` - присоединенное ИС для указанной команды.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция записывает присоединенное ИС для заданной команды режима управления (должен быть задан код команды и бит "прием/передача"). ВНИМАНИЕ! После вызова этой функции значение текущего подадреса не определено.

### **rtputflags**

```
void rtputflags(void far *pcBuffer, unsigned rtDir, unsigned  
rtFlagMin, unsigned rtFlagMax);
```

Параметры:

pcBuffer - адрес массива в ОЗУ ПЭВМ с записываемым блоком флаговых слов;

rtDir - выбор подадресов приема или подадресов передачи:

RT\_RECEIVE - подадреса приема,

RT\_TRANSMIT - подадреса передачи;

rtFlagMin, rtFlagMax - диапазон подадресов, для которых должны записываться флаговые слова (1 - 30).

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция записывает в ДОЗУ блок флаговых слов для заданного диапазона подадресов выбранного ОУ. Если rtFlagMin==rtFlagMax, записывается одно флаговое слово. Если rtFlagMin>rtFlagMax, функция не выполняет никакой записи. ВНИМАНИЕ! После вызова этой функции значение текущего подадреса не определено.

### **rtputw**

```
void rtputw(unsigned rtAddr, unsigned rtData);
```

Параметры:

rtAddr - адрес внутри подадреса (0 - 31);

rtData - записываемое слово.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция записывает слово rtData в текущий подадрес ДОЗУ выбранного ОУ по адресу rtAddr.

### **rtreset**

```
int rtreset();
```

Параметры:

Нет.

Возвращаемое значение:

0 - функция выполнена успешно,

TMK\_BAD\_FUNC - устройство не поддерживает работу в режиме ОУ.

Выполняемые действия:

Функция производит инициализацию выбранного устройства с переводом его в режим ОУ. При этом выбирается нулевая страница ДОЗУ и сбрасываются все биты ОС, для их установки надо использовать функцию rtsetanswbits. Значение текущего подадреса после вызова rtreset неопределено. Если устройство позволяет программирование режимов работы ОУ, то первый вызов rtreset после вызова tmkconfig включает в ОУ режим использования аппаратного бита и режим приема групповых сообщений и выключает режим работы с флагами. Последующие вызовы rtreset не изменяют установленных на момент вызова режимов работы ОУ.

### **rtrestore [DOS]**

```
void rtrestore();
```

Параметры:

Нет.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция должна применяться только в конце пользовательских функций обработки прерываний ОУ при необходимости восстановления сохраненного при прерывании состояния драйвера и прервавшего ОУ. Функция восстанавливает сохраненный номер выбранного устройства, который был установлен в момент прерывания, а также сохраненный номер подадреса ОУ, вызвавшего прерывание.

### **rtsetanswbits**

```
void rtsetanswbits(unsigned rtSetBits);
```

Параметры:

rtSetBits - маска, задающая установку в 1 указанных бит ОС:

SREQ - бит "Запрос обслуживания подсистемы",

BUSY - бит "Подсистема занята",

SSFL - бит "Неисправность подсистемы",

RTFL - бит "Неисправность терминала",

DNBA - бит "Готов принять управление каналом".

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция устанавливает заданные биты ответных слов, выдаваемых выбранным ОУ.

### **rtsetflag**

```
void rtsetflag();
```

Параметры:

Нет.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция устанавливает флаг в флаговом слове текущего подадреса выбранного ОУ в режиме работы ОУ с флагами.

### **rtunlock**

```
void rtunlock();
```

Параметры:

Нет.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция разблокирует заблокированный ранее функцией rtlock текущий подадрес выбранного ОУ в режиме работы ОУ без флагов.

\*\*\*\*\*

## **Функции для Многоадресного ОУ (устройства TX6/TE6/TA1-32RT)**

\*\*\*\*\*

### **mrtconfig [Win, Linux, QNX6]**

```
unsigned long mrtconfig(int mrtNumber);
```

Параметры:

mrtNumber - номер конфигурируемого устройства, начиная с 0;

Возвращаемое значение:

младшие 16 бит - минимальный номер БОУ, подключенного при вызове функции,

старшие 16 бит - количество BOY, подключенных при вызове функции;  
или

0 - ошибка при выполнении функции.

Выполняемые действия:

Функция подключает к вызвавшему процессу все BOY, находящиеся на физическом устройстве MOY с заданным номером. Если устройство с таким номером не существует или уже подключено к другому процессу, то возвращается 0.

### **mrtdefbrsubaddr0 [Win, Linux, QNX6]**

```
void mrtdefbrsubaddr0();
```

Параметры:

Нет.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция настраивает выбранное BOY, MOY и драйвер на дальнейшую работу с ДОЗУ в подадресе для приема групповых сообщений в выбранном MOY. Функция может быть использована только для устройств TX6/TE6.

### **mrtdefbrpage [Win, Linux, QNX6]**

```
int mrtdefbrpage(unsigned mrtBrcPage);
```

Параметры:

mrtBrcPage - выбираемая страница групповых сообщений ДОЗУ.

Возвращаемое значение:

0 - функция выполнена успешно;

RT\_BAD\_PAGE - задан недопустимый номер страницы групповых сообщений ДОЗУ.

Выполняемые действия:

Функция настраивает в выбранном BOY страницу групповых сообщений mrtBrcPage выбранного MOY в качестве текущей. В версии драйвера 4.00 используется только одна страница групповых сообщений с номером 0. При вызове с недопустимым для выбранного ОУ значением номера страницы возникает ошибочная ситуация с кодом ошибки RT\_BAD\_PAGE. Функция может быть использована только для устройств TA1-32RT.

### **mrtgetbrpage [Win, Linux, QNX6]**

```
unsigned mrtgetbrpage();
```

Параметры:

Нет.

Возвращаемое значение:

Номер текущей страницы групповых сообщений.

Выполняемые действия:

Функция возвращает заданный ранее функцией mrtdefbrpage номер страницы групповых сообщений в выбранном BOY. Функция может быть использована только для устройств TA1-32RT.

### **mrtgetmaxn [Win, Linux, QNX6]**

```
int mrtgetmaxn();
```

Параметры:

Нет.

Возвращаемое значение:

Максимальный возможный номер устройства MOY для данного драйвера.

Выполняемые действия:

Функция возвращает максимальный номер устройства, который может задаваться в функции mrtconfig. При наличии МОУ функция tmkgetmaxn возвращает максимальный существующий номер ВОУ, в отличие от mrtgetmaxn.

#### **mrtgetstate [Win, Linux, QNX6]**

`unsigned mrtgetstate();`

Параметры:

Нет.

Возвращаемое значение:

Слово состояния МОУ.

Выполняемые действия:

Функция читает слово состояния выбранного МОУ, содержащее биты последнего КС, принятого МОУ, бит занятости подадреса, бит ошибки сообщения МК.

#### **mrtreset [Win, Linux, QNX6]**

`int mrtreset();`

Параметры:

Нет.

Возвращаемое значение:

0 - функция выполнена успешно,

TMK\_BAD\_FUNC - устройство не является устройством МОУ.

Выполняемые действия:

Функция производит инициализацию выбранного устройства МОУ. Вызов mrtreset включает в МОУ режим использования аппаратного бита и режим приема групповых сообщений и выключает режим работы с флагами.

#### **mrtselected [Win, Linux, QNX6]**

`int mrtselected();`

Параметры:

Нет.

Возвращаемое значение:

Номер выбранного устройства МОУ.

Выполняемые действия:

Функция возвращает номер МОУ, в состав которого входит ВОУ, выбранное ранее в функциях tmkconfig или tmkselect.

\*\*\*\*\*

### **Функции общего назначения**

\*\*\*\*\*

#### **TmkOpen [Win, Linux, QNX6]**

`int TmkOpen();`

Параметры:

Нет.

Возвращаемое значение:

0 - функция выполнена успешно,

VTMK\_BAD\_VERSION - несовместимая версия драйвера, ненулевой код ошибки ОС.

Выполняемые действия:

Функция подключает драйвер к вызвавшему функцию процессу.

#### **TmkClose [Win, Linux, QNX6]**

`void TmkClose();`

Параметры:

Нет.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция отключает драйвер от вызвавшего функцию процесса.

### **tmkclrcwbits**

```
void tmkclrcwbits(unsigned tmkClrCWBits);
```

Параметры:

tmkClrCWBits - маска, задающая сброс в 0 указанных пользовательских бит ТМК: для устройств ТМК400, ТМКМРС, RTМК400 - биты CWB0 и CWB1 (соответствуют битам 5 и 6 Регистра Управляющего Слова; см. техническое описание устройств).

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция очищает заданные пользовательские биты выбранного ТМК. При вызове функций breset и rreset пользовательские биты обнуляются. Использование функции имеет значение только для устройств, в которых указанные биты задействованы аппаратно. В стандартных устройствах эти биты не используются.

### **tmkconfig [DOS]**

```
int tmkconfig(int tmkNumber, unsigned tmkType,  
              unsigned tmkPorts1, unsigned tmkPorts2,  
              char tmkIrq1, char tmkIrq2);
```

Параметры:

tmkNumber - номер конфигурируемого устройства, начиная с 0;

tmkType - тип установленного устройства:

ТМК400 - (соответствует устройству ТМК-400) устройство без резервирования ЛПИ, 8К слов ДОЗУ, программное определение установленного адреса ОУ, два базовых адреса портов (стандартно 0x100, 0x110), одна линия запроса прерывания (стандартно IRQ10);

ТМКМРС - (соответствует устройству МТМК-400 или ТМК-МР) устройство без резервирования ЛПИ, 2К слов ДОЗУ, невозможно программное задание и определение адреса ОУ, один базовый адрес портов (стандартно 0x100), одна линия запроса прерывания (стандартно IRQ5);

RTМК400 - (соответствует устройству РТМК-400) устройство с резервированием ЛПИ, 2К или 8К слов ДОЗУ, программное определение установленного адреса ОУ, программное задание режимов работы ОУ, один базовый адрес порта (стандартно 0x100), одна линия запроса прерывания (стандартно IRQ10);

ТМКХ - (соответствует всем ISA устройствам серий TX1, TX6, TE1, TE6, АМКО) устройство с резервированием ЛПИ, 2К или 8К или 16К слов ДОЗУ, программное задание режимов работы ОУ (кроме режима использования аппаратного бита), один базовый адрес порта (стандартно 0x160), одна линия запроса прерывания (стандартно IRQ10);

ТМКХI - (соответствует всем PCI устройствам серий TE1, TE6) устройство с резервированием ЛПИ, 16К слов ДОЗУ, программное задание режимов работы ОУ (кроме режима использования аппаратного бита);

ТА - (соответствует всем ISA устройствам серий ТА1, МПИ-ISA) устройство с резервированием ЛПИ, 64К слов ДОЗУ, программное задание режимов работы ОУ, один базовый адрес порта (стандартно 0x140), одна линия запроса прерывания (стандартно IRQ11);

ТАI - (соответствует всем PCI устройствам серии ТА1) устройство с резервированием ЛПИ, 64К слов ДОЗУ, программное задание режимов работы ОУ;

tmkPorts1 - первый базовый адрес портов для устройств ISA или номер платы PCI данного типа в ПК для плат PCI;

tmkPorts2 - второй базовый адрес портов для устройств ISA или номер устройства на плате PCI с несколькими устройствами;  
tmkIrq1 - номер первой линии запроса прерывания (0 - 15) устройств ISA, для устройств PCI не используется;  
tmkIrq2 - номер второй линии запроса прерывания (0 - 15) - начиная с версии драйвера 4.50 не используется.

Возвращаемое значение:

0 - конфигурирование драйвера выполнено,  
TMK\_BAD\_NUMBER - задан недопустимый номер устройства,  
TMK\_BAD\_TYPE - задан недопустимый тип устройства,  
TMK\_BAD\_IRQ - задан недопустимый номер линии прерывания,  
TMK\_PCI\_ERROR - неправильная попытка сконфигурировать устройство PCI.

Выполняемые действия:

Функция настраивает драйвер на конкретные значения адресов портов и номера линий прерывания для устройства с указанным номером. Неиспользуемые параметры (например, номера второй линии запроса прерывания для устройств, использующих одну линию) могут принимать значение 0xFFFF (для unsigned) или 0xFF (для char). Вызов функции с недопустимыми значениями параметров вызовет возникновение ошибочных ситуаций с кодами TMK\_BAD\_NUMBER, TMK\_BAD\_TYPE, TMK\_BAD\_IRQ. После выполнения этой функции конфигурируемое устройство остается выбранным для работы.

### **tmkconfig [Win, Linux, QNX6]**

```
int tmkconfig(int tmkNumber);
```

Параметры:

tmkNumber - номер конфигурируемого устройства, начиная с 0;

Возвращаемое значение:

0 - конфигурирование устройства выполнено,  
TMK\_BAD\_NUMBER - задан недопустимый номер устройства.

Выполняемые действия:

Функция подключает к вызвавшему процессу устройство с заданным номером. Если устройство с таким номером не существует или уже подключено к другому процессу, то возвращается код ошибки. После выполнения этой функции конфигурируемое устройство остается выбранным для работы.

### **tmkdeferrors [DOS]**

```
void tmkdeferrors(void (far* UserErrors)());
```

Параметры:

UserErrors - пользовательская функция, вызываемая при обнаружении ошибки при работе драйвера (при возникновении ошибочной ситуации); функция должна быть описана как far-функция, возвращающая тип void, и не имеющая входных параметров; код ошибки при этом содержится в переменной tmkError.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция задает драйверу имя пользовательской функции, которая будет вызываться каждый раз при обнаружении ошибки при работе драйвера. Функция предназначена только для использования при отладке программ, назначение пользовательской функции - вывод соответствующего сообщения.

### **tmkdefevent [Win]**

```
void tmkdefevent(HANDLE hEvent, BOOL fEventSet);
```

Параметры:

hEvent - событие Windows, созданное ранее вызовом Win32 API CreateEvent, нулевое значение отменяет заданное ранее событие;

fEventSet - должно быть равно TRUE.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция задает (или отменяет) для выбранного устройства событие Windows, которое драйвер будет использовать для информирования процесса о получении прерываний от устройства. Для получения непосредственной информации о причине прерываний должна использоваться функция tmkgetevd.

### **tmkdefirq [DOS]**

```
int tmkdefirq(char pcIrq);
```

Параметры:

pcIrq - номер линии запроса пользовательского прерывания (0 - 15).

Возвращаемое значение:

0 - функция выполнена успешно;

TMK\_BAD\_IRQ - задан недопустимый номер линии запроса прерывания.

Выполняемые действия:

Функция позволяет задать дополнительные прерывания, использующиеся в конкретных системах, которые должны маскироваться вместе с маскированием прерываний от устройств ТМК при прерываниях от устройств ТМК и при вызовах функции tmksave.

### **tmkdone**

```
void tmkdone(int tmkNumber);
```

Параметры:

tmkNumber - номер устройства ТМК, работу с которой драйвер заканчивает (для задания всех устройств можно использовать константу ALL\_TMKS).

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция должна вызываться в конце работы с устройствами ТМК. Основное назначение функции в DOS - восстановление системы прерываний ПЭВМ.

### **tmkgetcwbits**

```
unsigned tmkgetcwbits();
```

Параметры:

Нет.

Возвращаемое значение:

Состояние пользовательских битов ТМК: для устройств ТМК400, ТМКМРС, RTMK400 - биты CWB0 и CWB1 (соответствуют битам 5 и 6 Регистра Управляющего Слова; см. техническое описание устройств).

Выполняемые действия:

Функция возвращает текущее состояние пользовательских битов выбранного ТМК, заданных ранее функциями tmksetcwbits и tmkclrcwbits. При вызове функций breset и rreset пользовательские биты обнуляются. Использование функции имеет значение только для устройств, в которых указанные биты задействованы аппаратно. В стандартных устройствах эти биты не используются.

### **tmkgetevd [Win, Linux, QNX6]**

```
void tmkgetevd(TTmkEventData *pEvD);
```

Параметры:

pEvD - указатель на существующую структуру TTmkEventData.

Возвращаемое значение:



Нет.

Выполняемые действия:

Функция заполняет структуру, находящуюся по указанному адресу, данными очередного полученного от выбранного устройства прерывания.

### **tmkgetmaxn**

```
int tmkgetmaxn();
```

Параметры:

Нет.

Возвращаемое значение:

Максимальный возможный номер устройства ТМК для данного драйвера.

Выполняемые действия:

Функция возвращает максимальный номер устройства, который может задаваться в функциях `tmkconfig` и `tmkselect`.

### **tmkgetmode**

```
unsigned tmkgetmode();
```

Параметры:

Нет.

Возвращаемое значение:

Режим работы устройства:

BC\_MODE - режим Контроллера Канала;

RT\_MODE - режим Оконечного Устройства;

MT\_MODE - режим Монитора;

UNDEFINED\_MODE - режим не был задан.

Выполняемые действия:

Функция возвращает значение, описывающее режим работы выбранного устройства ТМК.

### **tmkgetevtime [Win]**

```
unsigned long tmkgetevtime();
```

Параметры:

Нет.

Возвращаемое значение:

Двойное слово (32 бита) зарегистрированного программного времени события.

Выполняемые действия:

Функция позволяет прочитать значение времени (значение программного таймера) регистрации последнего события, прочитанного функцией `tmkgetevd`.

Функция недоступна при работе с TA1-USB.

### **tmkgethwver**

```
unsigned tmkgethwver();
```

Параметры:

Нет.

Возвращаемое значение:

Аппаратная версия прошивки текущего выбранного устройства ТМК.

Выполняемые действия:

Функция позволяет узнать версию прошивки текущего выбранного устройства ТМК, прочитанную в момент запуска драйвера.

### **tmkgetswtimer [Win]**

```
unsigned long tmkgetswtimer();
```

Параметры:

Нет.

Возвращаемое значение:

Двойное слово (32 бита) текущего значения программного таймера.

Выполняемые действия:

Функция позволяет прочитать текущее значение программного таймера драйвера.

Функция недоступна при работе с TA1-USB.

### **tmkgettimer**

```
unsigned long tmkgettimer();
```

Параметры:

Нет.

Возвращаемое значение:

Двойное слово (32 бита) текущего значения аппаратного таймера.

Выполняемые действия:

Функция позволяет прочитать текущее значение аппаратного таймера выбранного устройства ТМК.

### **tmkgettimerl**

```
unsigned tmkgettimerl();
```

Параметры:

Нет.

Возвращаемое значение:

Младшее слово (16 бит) текущего значения аппаратного таймера.

Выполняемые действия:

Функция позволяет прочитать текущее значение младшего слова аппаратного таймера выбранного устройства ТМК.

### **tmkiodelay [DOS]**

```
unsigned tmkiodelay(unsigned IODelay);
```

Параметры:

IODelay - задержка ввода-вывода. Только для чтения текущего значения можно задать константу GET\_IO\_DELAY.

Возвращаемое значение:

Значение задержки ввода-вывода на момент вызова функции.

Выполняемые действия:

Функция задает задержку операций ввода-вывода для устройств ТМК400, RTМК400, ТМКМРС (см. файл pentium.doc) и позволяет прочитать текущее значение задержки.

### **tmkrestore [DOS]**

```
void tmkrestore();
```

Параметры:

Нет.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция восстанавливает из внутренних переменных драйвера данные, сохраненные вызовом tmksave: номер выбранного устройства ТМК и его состояние: номер базы для КК, номер подадреса (и его режим блокировки) для ОУ. Также функция размаскирует все прерывания устройств ТМК и прерывания, заданные функцией tmkdefirq. Основное назначение функции - использование в паре с функцией tmksave в пользовательских функциях обработки дополнительных (к прерываниям устройств ТМК) пользовательских прерываний, когда необходимо временно сохранять и восстанавливать состояние драйвера и маскировать прерывания устройств ТМК. Однако возможно применение пары этих функций и при обычном ходе выполнения программы для решения указанных задач.

### **tmksave [DOS]**

```
void tmksave();
```

Параметры:

Нет.

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция сохраняет во внутренних переменных драйвера номер выбранного устройства ТМК и его состояние: номер базы для КК, номер подадреса (и его режим блокировки) для ОУ, - а также маскирует все прерывания устройств ТМК и прерывания, заданные функцией `tmkdefirq`. Основное назначение функции - использование в паре с функцией `tmkrestore` в пользовательских функциях обработки дополнительных (к прерываниям устройств ТМК) пользовательских прерываний, когда необходимо временно сохранять и восстанавливать состояние драйвера и маскировать прерывания устройств ТМК. Однако возможно применение пары этих функций и при обычном ходе выполнения программы для решения указанных задач. При этом недопустимо вложение нескольких вызовов `tmksave`, а также вызов `tmksave` при обработке стандартных прерываний драйвера (в функциях `UserNormBC`, `UserExcBC`, `UserCmdRT`, `UserErrRT`), так как при этом прежние сохраненные значения будут потеряны.

### **tmkselect**

```
int tmkselect(int tmkNumber);
```

Параметры:

`tmkNumber` - номер выбираемого для работы устройства ТМК.

Возвращаемое значение:

0 - функция выполнена успешно;

`TMK_BAD_NUMBER` - задан недопустимый номер устройства;

`TMK_BAD_TYPE` - устройство не было сконфигурировано.

Выполняемые действия:

Функция оставляет в качестве выбранного устройство ТМК с заданным номером. Задание недопустимого номера приведет к возникновению ошибочной ситуации с кодом ошибки `TMK_BAD_NUMBER`. Если до вызова `tmkselect` устройство не было сконфигурировано функцией `tmkconfig`, возникнет ошибочная ситуация с кодом ошибки `TMK_BAD_TYPE`.

### **tmkselected**

```
int tmkselected();
```

Параметры:

Нет.

Возвращаемое значение:

Номер выбранного устройства.

Выполняемые действия:

Функция возвращает номер выбранного ранее в функциях `tmkconfig` или `tmkselect` устройства ТМК.

### **tmksetcwbits**

```
void tmksetcwbits(unsigned tmkSetCWBits);
```

Параметры:

`tmkSetCWBits` - маска, задающая установку в 1 указанных пользовательских бит ТМК: для устройств `TMK400`, `TMKMPC`, `RTMK400` - биты `CWB0` и `CWB1` (соответствуют битам 5 и 6 Регистра Управляющего Слова; см. техническое описание устройств).

Возвращаемое значение:

Нет.

Выполняемые действия:

Функция устанавливает заданные пользовательские биты выбранного ТМК. При вызове функций `bcreset` и `rtreset` пользовательские биты обнуляются. Использование функции

имеет значение только для устройств, в которых указанные биты задействованы аппаратно. В стандартных устройствах эти биты не используются.

### **tmkswtimer [Win]**

```
unsigned tmkswtimer(unsigned tmkTimerCtrl);
```

Параметры:

tmkTimerCtrl - параметры настройки программного таймера или операция функции.

Параметры настройки задаются комбинацией констант:

TIMER\_32BIT - работа в 32-битном режиме,

TIMER\_1US - дискретность 1 мкс.

Операция задается одной из констант:

SWTIMER\_OFF - выключение таймера,

SWTIMER\_ON - включение таймера (эквивалент TIMER\_32BIT | TIMER\_1US),

GET\_SWTIMER\_CTRL - получение текущих настроек таймера.

Возвращаемое значение:

Текущее значение настроек таймера - если драйвер поддерживает программный таймер и он включен;

0 - если драйвер не поддерживает программный таймер или таймер выключен.

Выполняемые действия:

Функция позволяет включать/выключать/настраивать программный таймер в драйвере.

Функция недоступна при работе с TA1-USB.

### **tmktimeout**

```
unsigned tmktimeout(unsigned tmkTimeOut);
```

Параметры:

tmkTimeOut - значение таймаута ожидания ОС или операция функции получения текущего значения GET\_TIMEOUT.

Значение таймаута задается обычным числом в микросекундах, драйвер подбирает ближайшее не меньшее аппаратно доступное значение таймаута.

Возвращаемое значение:

Значение настройки таймаута в микросекундах. Если аппаратура устройства не позволяет задать или получить значение таймаута, то возвращается 0.

Выполняемые действия:

Функция позволяет задать/узнать значение таймаута ожидания ОС на тех устройствах, где такая возможность аппаратно поддерживается.

### **tmktimer**

```
unsigned tmktimer(unsigned tmkTimerCtrl);
```

Параметры:

tmkTimerCtrl - параметры настройки аппаратного таймера или операция функции.

Параметры настройки задаются комбинацией констант:

TIMER\_16BIT - работа в 16-битном режиме,

TIMER\_32BIT - работа в 32-битном режиме,

TIMER\_1US - дискретность 1 мкс,

TIMER\_2US - дискретность 2 мкс,

TIMER\_4US - дискретность 4 мкс,

TIMER\_8US - дискретность 8 мкс,

TIMER\_16US - дискретность 16 мкс,

TIMER\_32US - дискретность 32 мкс,

TIMER\_64US - дискретность 64 мкс,

TIMER\_STOP - останов,

TIMER\_SYN - обнуление по команде "Синхронизация" (только в режиме ОУ),

TIMER\_SYND - обновление младших 16 бит таймера по команде "Синхронизация с СД" (только в режиме ОУ),

1...30 - обновление 32 бит таймера по приему данных в указанный подадрес (только в режиме ОУ).

Операция задается одной из констант:

TIMER\_RESET - сброс таймера и продолжение работы с текущими настройками,

TIMER\_OFF - сброс и выключение таймера,

GET\_TIMER\_CTRL - получение текущих настроек таймера.

Возвращаемое значение:

Текущее значение настроек таймера - если устройство поддерживает аппаратный таймер и он включен;

0 - если устройство не поддерживает аппаратный таймер или таймер выключен.

Выполняемые действия:

Функция позволяет включать/выключать/настраивать аппаратный таймер на выбранном устройстве TMK. Аппаратный таймер есть только на устройствах TA, TAI, MRTA, MRTAI.

### **tmkundefirq [DOS]**

```
int tmkundefirq(char pcIrq);
```

Параметры:

pcIrq - номер линии запроса пользовательского прерывания (0 - 15).

Возвращаемое значение:

0 - функция выполнена успешно;

TMK\_BAD\_IRQ - задан недопустимый номер линии запроса прерывания.

Выполняемые действия:

Функция отменяет управление линией запроса прерывания с номером pcIrq, заданное ранее в функции tmkdefirq.

### **tmkwaitevents [Linux, QNX6]**

```
int tmkwaitevents(int mask, int wait);
```

Параметры:

mask - битовая маска ожидаемых прерываний, биты с 0 по 30 соответствуют прерываниям от устройств с номерами с 0 по 30;

wait - время ожидания прерываний в миллисекундах, 0 - немедленный выход без ожидания, -1 - бесконечное ожидание.

Возвращаемое значение:

Отрицательное значение - код ошибки,

Положительное значение - битовая маска полученных прерываний,

Нулевое значение - нет прерываний.

Выполняемые действия:

Функция ожидает прихода заданных через параметр mask прерываний в течение времени, заданного через параметр wait. В маске прерываний можно задавать только прерывания устройств, захваченных ранее вызовами tmkconfig.

\*\*\*\*\*

### **Макросы**

\*\*\*\*\*

CW(ADDR,DIR,SUBADDR,NWORDS) - формирование KC обмена данными

CWM(ADDR,COMMAND) - формирование KC режима управления с признаком 11111

CWMC(ADDR,CI,COMMAND) - формирование KC режима управления с признаком, задаваемым CI

\*\*\*\*\*

### **Константы**

\*\*\*\*\*

Максимальный и минимальный поддерживаемый тип ТМК:

MAX\_TMK\_TYPE  
MIN\_TMK\_TYPE

Типы устройств ТМК:

TMK400  
TMKMPC  
RTMK400  
TMKX  
TMKXI  
TA  
TAI  
MRTA  
MRTAI

Режимы работы устройств ТМК:

BC_MODE	режим КК
RT_MODE	режим ОУ
MT_MODE	режим МТ
UNDEFINED_MODE	режим не был задан

Константа выбора всех устройств при вызове tmkdone

ALL\_TMKs

Пользовательские биты ТМК:

CWB0  
CWB1

Форматы сообщений в МК (коды управления для КК):

DATA_BC_RT	формат КК->ОУ (формат 1)
DATA_BC_RT_BRCST	формат КК->ОУ, групповой режим (формат 7)
DATA_RT_BC	формат ОУ->КК (формат 2)
DATA_RT_RT	формат ОУ->ОУ (формат 3)
DATA_RT_RT_BRCST	формат ОУ->ОУ, групповой режим (формат 8)
CTRL_C_A	формат КС - ОС (формат 4)
CTRL_C_BRCST	формат КС, групповой режим (формат 9)
CTRL_CD_A	формат КС+ИС - ОС (формат 6)
CTRL_CD_BRCST	формат КС+ИС, групповой режим (формат 10)
CTRL_C_AD	формат КС - ОС+ИС (формат 5)
или	
CC_FMT_1	формат КК->ОУ (формат 1)
CC_FMT_2	формат ОУ->КК (формат 2)
CC_FMT_3	формат ОУ->ОУ (формат 3)
CC_FMT_4	формат КС - ОС (формат 4)
CC_FMT_5	формат КС - ОС+ИС (формат 5)
CC_FMT_6	формат КС+ИС - ОС (формат 6)
CC_FMT_7	формат КК->ОУ, групповой режим (формат 7)
CC_FMT_8	формат ОУ->ОУ, групповой режим (формат 8)

CC_FMT_9	формат КС, групповой режим (формат 9)
CC_FMT_10	формат КС+ИС, групповой режим (формат 10)

Выбор ЛПИ в КК с резервированием:

BUS\_A  
BUS\_B

или

BUS\_1  
BUS\_2

Маски битов слова результата обмена КК [DOS]:

ERAO_MASK	ошибка адреса ОУ в ОС
MEO_MASK	ошибка кода "Манчестер-2"
IB_MASK	установлен бит в ОС
TO_MASK	ошибка паузы до ОС (нет ответа) или до любого ИС (нет или мало ИС)
EM_MASK	ошибка обмена с ДОЗУ (только на ТМК400, RTМК400, ТМКМРС)
EBC_MASK	ошибка самоконтроля при передаче или составная ошибка при приеме

Маски битов слова результата обмена КК [Win, Linux, QNX6]:

S_ERAO_MASK	ошибка адреса ОУ в ОС
S_MEO_MASK	ошибка кода "Манчестер-2"
S_IB_MASK	установлен бит в ОС
S_TO_MASK	ошибка паузы до ОС (нет ответа) или до любого ИС (нет или мало ИС)
S_EM_MASK	ошибка обмена с ДОЗУ (только на ТМК400, RTМК400, ТМКМРС)
S_EBC_MASK	ошибка самоконтроля при передаче или составная ошибка при приеме

Маски полей и битов командных и ответных слов:

NWORDS_MASK	поле числа слов
CMD_MASK	команда режима управления
SUBADDR_MASK	поле подадреса
CI_MASK	задание КС режима управления (все 1 в поле подадреса)
HBIT_MASK	аппаратный бит
RT_DIR_MASK	бит направления передачи
ADDRESS_MASK	поле адреса ОУ
RTFL_MASK	бит ОС "Неисправность ОУ"
DNBA_MASK	бит ОС "Принято управление интерфейсом"
SSFL_MASK	бит ОС "Неисправность подсистемы"
BUSY_MASK	бит ОС "Подсистема занята"
BRCST_MASK	бит ОС "Принята групповая команда"
NULL_MASK	резервные биты ОС
SREQ_MASK	бит ОС "Запрос на обслуживание подсистемы"
ERROR_MASK	бит ОС "Ошибка в сообщении"

Маски задания бит ОС в ОУ (rtsetanswbits, rtclranswbits, rtgetanswbits):

SREQ	задает бит ОС "Запрос на обслуживание подсистемы"
BUSY	задает бит ОС "Подсистема занята"
SSFL	задает бит ОС "Неисправность подсистемы"

RTFL	задает бит ОС "Неисправность ОУ"
DNBA	задает разрешение установки бита ОС "Принято управление интерфейсом"

Значения бита КС "прием/передача":

RT_TRANSMIT	ОУ передает
RT_RECEIVE	ОУ принимает

Бит "Ошибка сообщения МК" слова состояния ОУ:

RT\_ERROR\_MASK

Маски бита флага во флаговом слове ОУ:

RT\_FLAG  
RT\_FLAG\_MASK

Маски битов режимов работы ОУ:

RT_HBIT_MODE	включает режим использования аппаратного бита
RT_FLAG_MODE	включает режим использования флагов
RT_BRCST_MODE	включает режим разрешения приема групповых команд

Маски битов режимов прерываний ТМК

RT_DATA_BL	включает блокировку прерываний по приему/передаче данных ОУ
RT_GENER1_BL	включает блокировку прерываний ОУ по генерации по ЛПИ А ТМКХ
RT_GENER2_BL	включает блокировку прерываний ОУ по генерации по ЛПИ В ТМКХ
BC_GENER1_BL	включает блокировку прерываний КК по генерации по ЛПИ А ТМКХ
BC_GENER2_BL	включает блокировку прерываний КК по генерации по ЛПИ В ТМКХ
MT_GENER1_BL	включает блокировку прерываний МТ по генерации по ЛПИ А ТМКХ
MT_GENER2_BL	включает блокировку прерываний МТ по генерации по ЛПИ В ТМКХ
TMK_IRQ_OFF	отключает выход прерываний ISA устройств ТМКХ, ТА

Маски полей и битов расширенного кода управления

CX_CC_MASK	поле формата сообщения в МК
CX_CONT_MASK	бит продолжения цепочки
CX_BUS_MASK	бит задания ЛПИ
CX_SIG_MASK	бит сигнального прерывания
CX_INT_MASK	бит маскирования останова цепочки по ошибке (МТ)

Значения битов расширенного кода управления

CX_CONT	задает продолжение цепочки
CX_STOP	задает останов цепочки
CX_BUS_0	задает ЛПИ А
CX_BUS_A	задает ЛПИ А
CX_BUS_1	задает ЛПИ В
CX_BUS_B	задает ЛПИ В
CX_NOSIG	задает отсутствие сигнального прерывания
CX_SIG	задает сигнальное прерывание
CX_INT	задает останов цепочки по ошибке или установленному биту в ОС (МТ)
CX_NOINT	задает игнорирование ошибки или установленного бита в ОС (МТ)



## Маски полей и битов расширенного слова состояния

SX_ERR_MASK	поле расширенного кода ошибки
SX_IB_MASK	установлен бит в ОС
SX_G1_MASK	генерация по ЛПИ А
SX_G2_MASK	генерация по ЛПИ В
SX_K2_MASK	ошибка во 2-м ОС (MT)
SX_K1_MASK	ошибка в 1-м ОС (MT)
SX_SCC_MASK	поле кода формата сообщения (MT)
SX_ME_MASK	интегрированный признак ошибки в сообщении (MT)
SX_BUS_MASK	номер ЛПИ (MT)

## Значения расширенного кода ошибки

SX_NOERR	нет ошибки
SX_MEO	ошибка кода "Манчестер-2"
SX_TOA	ошибка паузы до ОС (нет ответа ОУ)
SX_TOD	ошибка паузы до ИС (число ИС меньше заданного)
SX_ELN	ошибка число ИС больше заданного
SX_EAO	ошибка адреса ОУ в ОС
SX_ESYN	ошибка типа синхроимпульса
SX_EBC	ошибка эхоконтроля при передаче или составная ошибка при приеме

## Номер ЛПИ в расширенном слове состояния

SX_BUS_0	ЛПИ А
SX_BUS_A	ЛПИ А
SX_BUS_1	ЛПИ В
SX_BUS_B	ЛПИ В

## Константа получения текущего значения задержки в tmkiodelay

GET\_IO\_DELAY

## Константы включения/выключения ОУ в rtenable

RT_ENABLE	включить ОУ
RT_DISABLE	выключить ОУ
RT_GET_ENABLE	получить состояние работы ОУ

## Константы управления программным таймером [Win]

SWTIMER_OFF	выключить программный таймер
SWTIMER_ON	включить программный таймер
TIMER_32BIT	разрядность таймера 32 бита
TIMER_1US	дискретность таймера 1 мкс
GET_SWTIMER_CTRL	получить состояние программного таймера

## Константы управления аппаратным таймером

TIMER_RESET	сбросить таймер и продолжить работу
TIMER_OFF	выключить таймер
TIMER_16BIT	разрядность таймера 16 бит

TIMER_32BIT	разрядность таймера 32 бита
TIMER_1US	дискретность таймера 1 мкс
TIMER_2US	дискретность таймера 2 мкс
TIMER_4US	дискретность таймера 4 мкс
TIMER_8US	дискретность таймера 8 мкс
TIMER_16US	дискретность таймера 16 мкс
TIMER_32US	дискретность таймера 32 мкс
TIMER_64US	дискретность таймера 64 мкс
TIMER_STOP	останов таймера
TIMER_SYN	обнуление таймера по команде “Синхронизация”
TIMER_SYND	обновление младшего слова таймера по команде “Синхронизация с СД”
TIMER_SA	маска задания поадреса обновления таймера
GET_TIMER_CTRL	получить состояние таймера

Константы управления таймаутом ожидания ОС

GET\_TIMEOUT           получить значение таймаута

Макросы формирования командных слов

CW(ADDR,DIR,SUBADDR,NWORDS) КС приема/передачи данных  
CWM(ADDR,COMMAND) КС режима управления  
CWMC(ADDR,CI,COMMAND) - КС режима управления с признаком, задаваемым CI

Коды команд режима управления

CMD_DYNAMIC_BUS_CONTROL	Принять управление интерфейсом
CMD_SYNCHRONIZE	Синхронизация
CMD_TRANSMIT_STATUS_WORD	Передать ОС
CMD_INITIATE_SELF_TEST	Начать самоконтроль ОУ
CMD_TRANSMITTER_SHUTDOWN	Блокировать передатчик
CMD_OVERRIDE_TRANSMITTER_SHUTDOWN	Разблокировать передатчик
CMD_INHIBIT_TERMINAL_FLAG_BIT	Блокировать признак неисправности ОУ
CMD_OVERRIDE_INHIBIT_TERMINAL_FLAG_BIT	Разблокировать признак неисправности ОУ
CMD_RESET_REMOTE_TERMINAL	Установить ОУ в исходное состояние
CMD_TRANSMIT_VECTOR_WORD	Передать векторное слово
CMD_SYNCHRONIZE_WITH_DATA_WORD	Синхронизация с СД
CMD_TRANSMIT_LAST_COMMAND_WORD	Передать последнюю команду
CMD_TRANSMIT_BUILT_IN_TEST_WORD	Передать слово ВСК ОУ

Коды ошибок:

TMK\_BAD\_TYPE  
TMK\_BAD\_IRQ  
TMK\_BAD\_NUMBER  
BC\_BAD\_BUS  
BC\_BAD\_BASE  
BC\_BAD\_LENGTH  
RT\_BAD\_PAGE  
RT\_BAD\_LENGTH  
RT\_BAD\_ADDRESS  
RT\_BAD\_FUNC

BC\_BAD\_FUNC  
TMK\_BAD\_FUNC  
TMK\_PCI\_ERROR

(с) ЗАО "Электронная компания "Элкус", 1995,2012.