v.2.02.

Руководство программиста по использованию: драйверной библиотеки TMKLL4 v7.xx (DOS), драйвера SYS tmk1553b.sys v4.xx (Windows NT 4.0), драйвера WDM 1553bwdm.sys v4.xx (Windows 98/ME/2000/XP/Vista/7), модуля драйвера tmk1553b.ko v4.xx (Linux kernel 2.2/2.4/2.6/3.0), администратора ресурса tmk1553b v4.xx (QNX 6.x).

Руководство программиста по использованию: драйвера WDM ezusb.sys и драйверных библиотек DLL USB_TA_DRV.dll, USB_TA_VC2.dll v1.x (Windows 98/ME/2000/XP/Vista/7), модуля драйвера tmk1553busb.ko v1.xx (Linux kernel 2.4/2.6/3.0) для модуля TA1-USB.

Основные сведения.

Терминал мультиплексного канала (ТМК) по ГОСТ Р 52070-2003 (ГОСТ 26765.52-87, MIL-STD-1553B) может работать в одном из трех режимов: контроллер канала (КК), оконечное устройство (ОУ) или монитор (МТ). Ранее разработанные устройства поддерживают работу только в режимах КК и ОУ. Более современные также могут работать в режиме МТ. Выбор режима осуществляется программно. Обмен данными с каналом происходит через двухпортовое буферное ОЗУ (ДОЗУ, БЗУ), доступ к которому возможен как со стороны ПК, так и со стороны мультиплексного канала (МК). Режим использования в МК аппаратного бита, в зависимости от типа устройства, может задаваться либо распайкой/установкой перемычки на устройстве либо программно; задание режима влияет только на работу ОУ и МТ (в режиме с аппаратным битом ОУ не воспринимает команды КК, в которых аппаратный бит равен 0).

Все драйвера предоставляют программисту единый интерфейс (хотя, небольшие различия для разных ОС все же существуют) для работы с разными типами ТМК и позволяет обойтись без непосредственного программирования устройств на уровне регистров. Это дает возможность писать практически универсальные программы, не зависящие от типа применяемой платы (за исключением тех случаев, когда используются какие-либо уникальные особенности выбранного типа плат).

DOS

Программный драйвер DOS низкого уровня TMKLL4 v7.xx реализован в виде библиотеки функций для языков C/C++. Возможно использование моделей памяти от TINY до LARGE. Предпочтительной моделью памяти является LARGE, так как все вызовы функций драйвера производятся по правилам именно этой модели. Для модели памяти HUGE объектный файл библиотеки должен быть перекомпилирован!

Linux

Основой драйвера tmk1553b плат терминалов мультиплексного канала для Linux является модуль драйвера tmk1553b.ko (для TA1-USB - tmk1553busb.ko). Модуль должен быть загружен с заданием необходимых параметров командой insmod и создан файл устройства /dev/tmk1553b (для TA1-USB - /dev/tmk1553busb0.../dev/tmk1553busb7) командой mknod в соответствии с файлом readme.

Один модуль драйвера управляет всеми устройствами ТМК, установленными в компьютер. При этом каждым отдельным устройством в каждый момент времени может управлять только один процесс, захвативший устройство вызовом tmkconfig.

QNX6

Основой драйвера tmk1553b плат терминалов мультиплексного канала для QNX6 является администратор ресурса tmk1553b. Администратор ресурса должен быть запущен с заданием необходимых параметров в соответствии с файлом readme.txt.

Один администратор ресурса управляет всеми устройствами ТМК, установленными в компьютер. При этом каждым отдельным устройством в каждый момент времени может управлять только один процесс, захвативший устройство вызовом tmkconfig.

Windows NT 4.0

Основой драйвера TMK1553B плат терминалов мультиплексного канала для Microsoft Windows NT 4.0 является драйвер tmk1553b.sys. Файл tmk1553b.sys должен находиться в директории $(WINDIR)\system32\drivers. Установка и запуск драйвера выполняются вручную в соответствии с файлом readwin.txt.

Один драйвер управляет всеми устройствами ТМК, установленными в компьютер, конфигурация устройств хранится в реестре Windows NT 4.0. При этом каждым отдельным устройством в каждый момент времени может управлять только один Win32 процесс, захвативший устройство вызовом tmkconfig.

Windows 98,ME,2000,XP,Vista,7

Основой драйвера TMK1553B плат терминалов мультиплексного канала для Microsoft Windows 98/ME/2000/XP/Vista/7 является драйвер 1553bwdm.sys (для TA1-USB - драйвер ezusb.sys и библиотеки функций USB_TA_DRV.dll и USB_TA_VC2.dll). Файл 1553bwdm.sys (ezusb.sys) должен находиться в директории $(WINDIR)\system32\drivers. Установка драйвера выполняется средствами Windows в соответствии с файлом readme.txt. Установленный драйвер запускается автоматически при старте Windows.

Для TA1-USB библиотеки функций USB_TA_DRV.dll и USB_TA_VC2.dll должны находиться в папке рабочего проекта.

Для каждого установленного в компьютер устройства создается уникальное символьное имя, к которому имеет доступ только один процесс, захвативший устройство вызовом tmkconfig.

All

Все драйвера поддерживают следующие типы устройств:

- TMK400 - нерезервированный ТМК, ДОЗУ 8Кх16 (128 баз в режиме КК, 4 страницы в режиме ОУ), одновременное переключение страниц для МК и ПК в режиме ОУ, программное задание адреса ОУ в МК, возможно чтение из устройства установленного адреса ОУ в МК;
- TMKMPC - нерезервированный ТМК, ДОЗУ 2Кх16 (32 базы в режиме КК, 1 страница в режиме ОУ), невозможно программное задание и определение адреса ОУ в МК (адрес задается перемычками на устройстве), плата выполнена в стандарте MicroPC фирмы Octagon Systems;
- RTMK400 - резервированный ТМК, ДОЗУ 2Кх16 на старых платах (32 базы в режиме КК, 1 страница в режиме ОУ), или 8Кх16 на новых платах (128 баз в режиме КК, 4 страницы в режиме ОУ), одновременное переключение страниц для МК и ПК в режиме ОУ,

возможность работы по одной из двух линий передачи информации (основной или резервной), программное задание адреса ОУ в МК, возможно чтение из устройства установленного адреса ОУ в МК, программное задание режимов работы ОУ (использование аппаратного бита в МК, режим работы с флагами или без флагов, использование команд с групповой адресацией в МК);
- TMKX - резервированный ТМК с режимом МТ, ДОЗУ 8Кх16 на старых платах ТХ1-РС (128 баз в режимах КК и МТ, 4 страницы в режиме ОУ) или 2Кх16 на старых платах ТХ1-МР, ТХ1-104 и на платах АМКО (32 базы в режиме КК и МТ, 1 страница в режиме ОУ) или 16Кх16 (256 баз в режимах КК и МТ, 8 страниц в режиме ОУ) на всех последних платах серий ТХ1, ТХ6, ТЕ1, ТЕ6, возможность работы по одной из двух линий передачи информации (основной или резервной), раздельное переключение страниц для МК и ПК в режиме ОУ, программное задание адреса ОУ в МК; к платам этого типа относятся все платы ТХ/ТЕ с шиной ISA, имеющие единый программный интерфейс и различающиеся только конструктивным исполнением и объемом установленного ОЗУ (объем установленного ОЗУ определяется драйвером автоматически при вызове функции tmkconfig): ТХ1-РС, ТХ6-РС, ТХ1-МР, ТХ1-104, ТЕ1-РС, ТЕ1-104, ТХ1-В, АМКО;
- TMKXI - аналог плат TMKX, но для шины PCI, ДОЗУ 16Кх16 (256 баз в режимах КК и МТ, 8 страниц в режиме ОУ); к платам этого типа относятся платы ТЕ1-PCI, ТЕ6-PCI, ТЕ1-PCI2, ТЕ6-PCI2, ТЕ1-3U-CPCI, ТЕ1-6U-CPCI;
- MRTX (кроме DOS) - резервированный ТМК на базе TMKX (платы серий ТХ6/ТЕ6) в режиме только многоадресного ОУ (4 ОУ);
- MRTXI (кроме DOS) - резервированный ТМК на базе TMKXI (платы серий ТХ6/ТЕ6) в режиме только многоадресного ОУ (4 ОУ);
- ТА - резервированный ТМК с режимом МТ, ДОЗУ 64Кх16 (1024 базы в режиме КК, 512 баз в режиме МТ, 1 страница в режиме ОУ), возможность работы по одной из двух линий передачи информации (основной или резервной), программное задание адреса ОУ в МК; к платам этого типа относятся все платы ТА1 с шиной ISA, имеющие единый программный интерфейс и различающиеся только конструктивным исполнением: ТА1-РС, ТА1-МР, ТА1-104, МПИ-ISA, - и микромодули ТАМ1; к этому же типу относятся и модули с интерфейсом USB - ТА1-USB;
- TAI - аналог плат ТА, но для шины PCI, ДОЗУ 64Кх16 (1024 базы в режиме КК, 512 баз в режиме МТ, 1 страница в режиме ОУ); к платам этого типа относятся платы ТА1-PCI, ТА1-PCI4, ТА1-104PCI, ТА1-PMC2, ТА1-3U-CPCI, ТА1-6U-CPCI;
- MRTA (кроме DOS) - резервированный ТМК на базе ТА в режиме многоадресного ОУ (32 ОУ), ДОЗУ 256Кх16, с шиной ISA; физически плат такого типа не существует;
- MRTAI (кроме DOS) - резервированный ТМК на базе TAI в режиме многоадресного ОУ (32 ОУ), ДОЗУ 256Кх16, с шиной PCI; к платам такого типа относятся ТА1-PCI-32RT.

Драйвер поддерживает работу с несколькими физическими устройствами ТМК одновременно. Количество поддерживаемых устройств задается при компиляции драйвера: 4 для DOS, 8 для Linux/QNX6/Windows (в принципе, возможна поставка драйвера той же версии для работы с другим количеством устройств). Если на реальной плате находится несколько устройств (например, ТЕ1-PCI2, ТХ1-РС-2 содержат по два устройства, ТА1-PCI4 - до четырех устройств), то каждая такая плата понимается как несколько ТМК. Расположенные на одной плате различные физические ТМК абсолютно независимы друг от друга и при работе никак не влияют друг на друга.

Устройства на USB нумеруются дополнительно к основным ISA/PCI устройствам, установленным внутри компьютера. Число устройств на USB также может быть до 8. При работе с драйвером нужно учитывать нумерацию устройств при выполнении функции TmkOpen(). Эта функция в начале определяет количество доступных ISA/PCI устройств, поддерживаемых драйвером. Затем определяется наличие подключенных USB устройств. Например, при установленной в компьютере плате ТА1-PCI и подключенном

модуле TA1-USB, нумерация USB устройств начнется с номера 1 в Windows и с номера 8 в Linux. При отсутствии установленных ISA/PCI плат с драйвером 1553bwdm.sys в Windows и отсутствии загруженного драйвера tmk1553b.ko в Linux нумерация устройств TA1-USB начнется с 0.

Все одновременно работающие ТМК могут быть разных типов и находиться в любом режиме - КК, МТ или ОУ. Смена режима ТМК также может происходить независимо и в любой момент времени.

Кроме того, для поддержки режима многоадресных ОУ введены виртуальные ТМК. Виртуальные ТМК реализуются на базе специальных физических ТМК, поддерживающих режим многоадресных ОУ. Это все платы серий TX6/TE6, а также плата TA1-PCI-32RT. Основная особенность виртуальных ТМК в том, что несколько виртуальных ТМК реализованы на базе одного и того же физического ТМК. По этой причине, есть некоторые ограничения по работе с такими ТМК, и существует определенное взаимовлияние работы каждого виртуального ТМК на своих соседей по родительскому физическому ТМК. Например, программное изменение какого-то режима в одном из виртуальных ТМК может привести к таким же изменениям у всех виртуальных соседних ТМК. Виртуальные ТМК могут работать только в режиме ОУ. Если физический ТМК сконфигурирован для работы в режиме многоадресного ОУ, он уже не может работать в стандартных режимах КК, ОУ, МТ. Помимо физических ТМК, драйвера Windows поддерживают дополнительно до 31 виртуальных ТМК, а драйвера Linux и QNX6 - до 23 виртуальных ТМК.

Конфигурация устройства для работы в качестве стандартного физического КК/ОУ/МТ или нескольких виртуальных ОУ осуществляется вручную при конфигурировании драйверов в операционных системах Windows, Linux, QNX6 или запуском соответствующей программы загрузчика плат в DOS. Эта конфигурация не может быть изменена программно через интерфейс программирования драйвера.

DOS

Драйвер программно настраивается на конкретную конфигурацию используемых ТМК в ПК (типы ТМК, используемые адреса портов ввода/вывода и линии прерываний). Вместе с драйвером поставляется файл tmkinit.c с функциями, позволяющими задать конфигурацию драйвера из текстового файла конфигурации.

Драйвер не позволяет разделять прерывания с другими программами, то есть, если устройство ТМК на шине PCI разделяет свое прерывание с какой-либо другой платой, и драйвер другой платы обслуживал ее прерывания, то после вызова функции tmkconfig чужие прерывания не будут транслироваться в предыдущий обработчик.

Linux, QNX6, Windows

Драйвер настраивается на конфигурацию используемых ТМК в ПК автоматически или вручную при запуске драйвера. Программа может лишь определить список доступных устройств и использовать все или часть из них.

Подключение драйвера в программах

DOS

Для использования драйвера необходимо директивой #include включить в текст программы заголовочный файл tmkll4.h

```
#include "tmkll4.h"
```

При работе в интегрированной среде (IDE) необходимо использовать файл проекта (.prj) с подключенным в нем драйвером tmkll4.obj. При работе из командной строки необходимо указывать драйвер в опции линкеру.

Linux

Для работы с драйвером необходимо с помощью директивы

```
#include "ltmk.c"
```

подключить в текст программы на языке C/C++ файл ltmk.c, содержащий определения всех констант и функций API драйвера tmk1553b.

Для поддержки TA1-USB (одновременно с устройствами ISA/PCI) файл ltmk.c должен браться из комплекта драйвера Linux TA1-USB.

QNX6

Для работы с драйвером необходимо с помощью директивы

```
#include "qnx6tmk.c"
```

подключить в текст программы на языке C/C++ файл qnx6tmk.c, содержащий определения всех констант и функций API драйвера tmk1553b.

Windows NT 4.0

Для работы с драйвером необходимо с помощью директивы

```
#include "ntmk.c"
```

подключить в текст программы на языке C/C++ файл ntmk.c, содержащий определения всех констант и функций API драйвера tmk1553b.sys.

Windows 98/ME/2000/XP/Vista/7

Для работы с драйвером необходимо с помощью директивы

```
#include "WDMTMKv2.cpp"
```

подключить в текст программы на языке C/C++ файл WDMTMKv2.cpp, содержащий определения всех констант и функций API драйвера 1553bwdm.sys.

Для поддержки TA1-USB (одновременно с устройствами ISA/PCI) файл WDMTMKv2.cpp должен браться из комплекта драйвера Windows TA1-USB. Дополнительно для TA1-USB необходимо библиотеки функций USB_TA_DRV.dll и USB_TA_VC2.dll разместить в папке рабочего проекта.

Начало и завершение работы с драйвером

DOS

Драйвер целиком встраивается в программу пользователя и не требует вызова специальных функций для начала и завершения работы.

Linux, QNX6, Windows

Драйвер является частью операционной системы, и программа, для получения доступа к аппаратуре, управляемой драйвером, должна использовать специальные функции для начала и окончания взаимодействия с драйвером.

Linux

Начинать работу с драйвером необходимо с вызова функции TmkOpen:

```
int TmkOpen();
```

которая открывает файл устройства и производит начальную настройку. При успешном завершении функция возвращает нулевое значение. При невозможности открыть драйвер вызовом open возвращается код ошибки.

После окончания работы с tmk1553b должна вызываться функция TmkClose:

```
void TmkClose();
```

QNX6

Начинать работу с драйвером необходимо с вызова функции TmkOpen:

```
DWORD TmkOpen();
```

которая открывает файл устройства и производит начальную настройку. При успешном завершении функция возвращает нулевое значение.

После окончания работы с tmk1553b должна вызываться функция TmkClose:

```
void TmkClose();
```

Windows NT 4.0

Начинать работу с драйвером необходимо с вызова функции TmkOpen:

```
DWORD TmkOpen();
```

которая динамически загружает драйвер в память и производит его начальную настройку. При успешном завершении функция возвращает нулевое значение. При невозможности загрузить драйвер вызовом CreateFile возвращается код ошибки, полученный вызовом GetLastError.

После окончания работы с tmk1553b должна вызываться функция TmkClose:

```
void TmkClose();
```

Windows 98/ME/2000/XP/Vista/7

Начинать работу с драйвером необходимо с вызова функции TmkOpen:

```
DWORD TmkOpen();
```

которая динамически загружает драйвер в память и производит его начальную настройку. При успешном завершении функция возвращает нулевое значение. При невозможности загрузить драйвер вызовом CreateFile возвращается код 1.

После окончания работы с 1553bwdm должна вызываться функция TmkClose:

```
void TmkClose();
```

Функции общего назначения

Каждое устройство, управляемое драйвером, имеет свой номер от 0 до максимального значения, поддерживаемого драйвером (для драйвера DOS этот диапазон равен 0 - 3, для остальных драйверов 0 - 7). Функция tmkgetmaxn возвращает указанное максимальное значение, являющееся константой для данного драйвера.

В начале работы программе необходимо перечислить устройства, с которыми она собирается работать, а в DOS еще и задать их конфигурацию в ПК. Это делается с помощью функции tmkconfig, вызываемой отдельно для каждого из устройств.

DOS

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

Функция ставит в соответствие номеру устройства определенный тип устройства, используемые адреса портов ввода/вывода ПК и используемые линии прерываний. При выполнении этой функции драйвер настраивается на работу с указанным типом устройства, а прерывания с заданными номерами перехватываются драйвером и размаскируются.

Linux, QNX6

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

Если устройство с заданным номером указано при запуске драйвера и с устройством к этому моменту не работает ни один процесс, то функция закрепляет это устройство за вызвавшим tmkconfig процессом, и в дальнейшем этот процесс получает исключительный доступ к устройству. При этом функция tmkconfig возвращает нулевое значение. Если устройство уже закреплено за каким-либо процессом или вообще не задано при запуске драйвера, то возвращается ненулевой код ошибки.

Windows

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

Если устройство с заданным номером описано в реестре Windows и с устройством к этому моменту не работает ни один процесс, то функция закрепляет это устройство за вызвавшим tmkconfig процессом, и в дальнейшем этот процесс получает исключительный доступ к устройству. При этом функция tmkconfig возвращает нулевое значение. Если устройство уже закреплено за каким-либо процессом или вообще не описано в реестре, то возвращается ненулевой код ошибки.

All

Программа (процесс) освобождает устройства, с которыми закончена работа, через вызов функции tmkdone:

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

В качестве параметра можно задавать либо номер конкретного устройства, либо константу ALL_TMKS - при этом освобождаются все устройства, заданные ранее в tmkconfig. В DOS tmkdone приводит также к восстановлению состояний перехваченных прерываний и их маскированию.

Все функции драйвера, работающие с отдельными устройствами (кроме tmkconfig и tmkdone), используют понятие текущего выбранного устройства, то есть в списке их параметров отсутствует номер устройства, для которого вызывается функция. Для выбора текущего устройства используется функция tmkselect, а для определения номера выбранного устройства - функция tmkselected. Функцией tmkselected, например, можно пользоваться внутри одного DOS обработчика прерываний от нескольких устройств для определения устройства - источника прерывания. Функция tmkconfig также оставляет выбранным то устройство, для которого вызывалась. Поэтому, в случае применения единственного устройства, его номер (любой от 0 до максимального) задается один раз при вызове tmkconfig. Для нескольких устройств потребуется их переключение с помощью tmkselect. При этом состояние невыбранных устройств не изменяется.

Режим КК, ОУ или МТ для выбранного устройства задается функциями bcreset, rtreset или mtreset, соответственно. Эти функции возвращают 0 в случае успешного завершения и значение TMK_BAD_FUNC, если выбранное устройство не поддерживает работу в заданном режиме. Функция tmkgetmode позволяет определить режим выбранного устройства и возвращает одно из трех значений: BC_MODE - режим КК, RT_MODE - режим ОУ, MT_MODE - режим МТ, UNDEFINED_MODE - режим устройства не был определен.

Для устройств TA, TAI, MRTA, MRTAI можно настраивать таймаут ожидания ответного слова. Для этого введена функция tmktimeout. В качестве параметра задается либо требуемое значение таймаута в мкс, либо константа GET_TIMEOUT для чтения текущего значения. Если задано значение таймаута, драйвер преобразует его к одному из аппаратно поддерживаемых значений, не меньше заданного. Функция возвращает текущее установленное значение в мкс или 0, если функция не поддерживается на текущем выбранном устройстве.

В устройствах TA, TAI, MRTA, MRTAI встроен аппаратный 32-битный таймер, исходно он выключен. Если его включить, то в момент завершения обработки сообщения в любом из режимов (КК/ОУ/МТ) текущее значение таймера будет сохранено в рабочей базе/подадресе сообщения. Таймер управляется функцией tmktimer. В качестве параметра надо задавать слово управления таймером, которое либо задает операцию, либо режим работы таймера. Существуют три операции - TIMER_OFF (сброс и выключение таймера), TIMER_RESET (сброс и продолжение работы таймера), GET_TIMER_CTRL (получение текущих настроек таймера). Режим работы таймера задается комбинацией по ИЛИ констант битности, дискретности, настроек обновления в режиме ОУ. Константы битности TIMER_16BIT или TIMER_32BIT задают 16 или 32 разрядный режим работы таймера. Константы дискретности задают шаг счета и значение младшего разряда таймера: 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. Настройки обновления таймера в режиме ОУ задаются константами: TIMER_SYN - обнуление по команде “Синхронизация”, TIMER_SYND - обновление младших 16 бит таймера по команде “Синхронизация с СД”, или число в диапазоне 1...30, задающее обновление 32 бит таймера по приему данных в указанный подадрес. Функция tmktimer возвращает текущее значение настроек таймера, если таковой поддерживается на выбранном устройстве, или 0 - если не поддерживается.

Текущее 32-битное значение таймера можно прочитать функцией tmkgettimer, а только младшие 16 бит могут быть прочитаны функцией tmkgettimerl. Для чтения значения таймера, сохраненного при обработке сообщения в КК/ОУ/МТ в текущей базе КК, в текущем подадресе ОУ, в текущей базе МТ - предназначены функции bcgetmsgtime, rtgetmsgtime, mtgetmsgtime соответственно.

Windows

В драйвере Windows дополнительно реализован программный 32-битный таймер с дискретностью 1 мкс. Управление таймером осуществляется функцией tmkswtimer, а чтение текущего значения - tmkgetswtimer. Таймер включается константой SWTIMER_ON (являющейся комбинацией TIMER_32BIT | TIMER_1US), а выключается константой SWTIMER_OFF. Текущий режим можно получить, задав константу GET_SWTIMER_CTRL. Если таймер включен, то при получении каждого прерывания драйвер сохраняет в программном буфере прерываний текущее значение программного таймера. Далее, сразу после получения данных прерывания функцией tmkgetevd (см. ниже), можно получить и значение таймера в момент получения прерывания драйвером - для этого предназначена функция tmkgetevtime.

Функции программного таймера Windows недоступны при работе с TA1-USB.

All

Некоторые устройства имеют в своих регистрах неиспользуемые разряды (2 бита для устройств TMK400, TMKMPC и RTMK400), которые аппаратно доступны на устройствах и могут использоваться в нестандартных модификациях устройств. Работа с этими разрядами поддерживается функциями tmksetcwbits, tmkclrcwbits и tmkgetcwbits, позволяющими, соответственно, устанавливать, очищать и определять текущее состояние пользовательских разрядов. Для выбора разрядов в этих функциях надо использовать предопределенные константы CWB0, CWB1 или их комбинацию. Вызовы функций bcreset и rtreset приводят к обнулению пользовательских разрядов.

Существует возможная проблема работы устройств TMK400, RTMK400 и TMKMPC на быстрой шине ISA, когда последовательные циклы по шине ISA могут генерироваться системой быстрее, чем их может выполнить аппаратура плат. В основной массе современных компьютеров такая проблема не возникает. Нами отмечены проблемы на самых первых чипсетах для процессоров Intel Pentium, AMD K5, Cyrix686+. В некоторых чипсетах скорость шины ISA регулируется через BIOS. Если такой регулировки нет, то уменьшить скорость генерации циклов обращения к платам можно с помощью функции tmkiodelay, позволяющую задать задержку при обращении к портам плат TMK400, RTMK400, TMKMPC. Для плат типа TMKX, TMKXI, TA, TAI задание задержек не требуется, а вызовы функции tmkiodelay для этих плат драйвером игнорируются.

Функция получает единственный параметр, задающий число циклов задержки в командах обращения к портам текущего выбранного устройства, а возвращает значение задержки, существовавшее на момент вызова этой функции. Диапазон допустимых значений задержки от 1 до 65534. Вызов tmkiodelay(0) эквивалентен вызову tmkiodelay(1). Значение 65535 зарезервировано за константой GET_IO_DELAY, задание которой в качестве параметра позволяет узнать текущие значения задержек без их изменения. Для каждого устройства могут быть заданы свои задержки.

DOS

Кроме того, в файл tmktest.c добавлена функция TMK_TuneIODelay, которая позволяет экспериментально подобрать задержку для текущего выбранного устройства, выполняя тесты ОЗУ устройства в режиме КК при различных значениях задержек.

Функция требует единственный параметр, задающий максимальное значение подбираемой задержки. Вызов TMK_TuneIODelay(0) использует максимальное значение задержки по умолчанию (50). Максимальное значение позволяет ограничить время подбора при вызове функции для отсутствующего (неправильно заданной в конфигурации) устройства.
В качестве результата функция возвращает 0 при успешно подобранной задержке или константу BC_BAD_RAM, если в заданном диапазоне задержек не удалось подобрать задержку, при которой плата бы работала без сбоев. После вызова TMK_TuneIODelay плата находится в режиме КК.

Linux, QNX6, Windows

Драйвера для Linux, QNX6 и Windows при запуске всегда автоматически вызывают внутренний вариант TMK_TuneIODelay для устройств TMK400, RTMK400, TMKMPC, поэтому программы всегда работают с оптимальными настройками скорости интерфейса.

All

При вызове функций драйвера обеспечивается контроль практически всех параметров, могущих повлиять на правильность работы функций. При этом выполняется одна из двух операций: либо значение корректируется по умолчанию маскированием всех "лишних" для данной функции разрядов, и далее функция выполняется с корректированным значением (при передаче таких параметров состояние их незначащей части может быть любым), либо значение, которое не может быть скорректировано таким образом, полностью проверяется на допустимость, и, если оно не удовлетворяет критериям правильности, возникает ошибочная ситуация, и функция аварийно завершается. Для обработки таких ошибочных ситуаций в драйвере приняты следующие правила.

DOS

Введена доступная программисту переменная tmkError, и каждая функция, в которой может возникнуть ошибочная ситуация, устанавливает эту переменную (функции, корректирующие параметры по умолчанию, не изменяют значение tmkError). Если ошибок не было, в tmkError записывается нулевое значение, в противном случае - ненулевой код ошибки. Кроме того, программист может задать с помощью функции драйвера tmkdeferrors свою функцию реакции на ошибку, которая будет вызываться драйвером всякий раз в момент обнаружения ошибки параметра. Такой вариант целесообразно использовать только при отладке программ, поскольку большинство ошибок вызывается недопустимыми параметрами функций из-за программных ошибок. Однако, есть ряд параметров, правильность которых зависит от текущей конфигурации. Функции с такими параметрами (конфигурирование, выбор устройства, задание базы КК/МТ, задание страницы ОУ и т.п.) кроме установки tmkError, возвращают в качестве результата значение, дублирующее tmkError.

Linux

И сам драйвер, и его интерфейс ltmk.c включают необходимый код для поддержки правильных значений переменной tmkError, аналогично драйверу DOS. Но, по умолчанию, вся работа с tmkError в ltmk.c отключена. В общем случае, предполагается только возможность анализа кода возврата тех функций, где значение tmkError дублируется виде кода возврата (конфигурирование, выбор устройства, задание базы КК/МТ, задание страницы ОУ и т.п.). При необходимости можно включить полную обработку tmkError в ltmk.c, раскомментировав строку #define USE_TMK_ERROR в файле ltmk.c. Функция tmkdeferrors в драйвере Linux не реализована.

QNX6, Windows

Интерфейсы драйверов QNX6 и Windows не содержат кода поддержки правильных значений переменной tmkError. Предполагается только возможность анализа кода возврата тех функций, где значение tmkError дублируется виде кода возврата (конфигурирование, выбор устройства, задание базы КК/МТ, задание страницы ОУ и т.п.). Функция tmkdeferrors в драйверах QNX6 и Windows не реализована.

Обработка прерываний

Во всех режимах устройство при работе генерирует прерывания. Каждое прерывание обрабатывается драйвером, а затем тем или иным образом информация о прерывании передается пользовательской программе. Механизм передачи информации о прерывании различен для разных ОС. В DOS драйвер прямо из своего обработчика прерываний вызывает пользовательский код, оформленный в виде специальных функций, таким образом, программист имеет возможность встроить свой код прямо в обработчик. В других ОС обработчик прерываний полностью встроен в код драйвера, выполняющийся на уровне ядра ОС, и не имеет возможности передать управление какому-либо пользовательскому коду в момент прерывания. Для уменьшения влияния задержек переключения на пользовательский код при обработке прерываний в многозадачных ОС драйвер включает в себя буфер на 512 прерываний.

Ниже перечислены типы прерываний, обрабатываемых драйвером, и данные, сопровождающие каждый тип прерывания. Названия типов прерываний (bcIntNorm и т.д.) в данном случае чисто условны и используются только в данном тексте.
bcIntNorm - прерывание нормального завершения одиночного обмена в режиме КК. Данные: wResult == 0 (слово результата обмена равно нулю).
bcIntExc - прерывание завершения одиночного обмена с исключительной ситуацией в режиме КК. Данные: wResult (слово результата обмена), wAW1, wAW2 (ответные слова).
bcIntX - прерывание завершение обмена (останова) КК. Данные: wResultX (расширенное слово результата обмена), wBase (база останова).
bcIntSig - сигнальное прерывание КК. Данные: wBase (текущая база).
rtIntCmd - прерывание по команде режима управления ОУ. Данные: wCmd (код команды режима управления по маске 0x41F).
rtIntErr - прерывание по ошибке обмена ОУ. Данные: wStatus (слово состояния ОУ).
rtIntData - прерывание по команде приема/передачи данных ОУ. Данные: wStatus (слово состояния ОУ).
mtIntX - прерывание останова МТ. Данные: wResultX (расширенное слово результата), wBase (база останова).
mtIntSig - сигнальное прерывание МТ. Данные: wBase (текущая база).
tmkIntOth - дополнительное прерывание ТМК, не привязанное к режиму работы (прерывание таймера, переполнение буфера прерываний). Данные: wRequest (слово запроса прерывания).

DOS

Перед началом работы программа должна зарегистрировать в драйвере свои обработчики прерываний для всех режимов и прерываний, которые планируется использовать. Вот пример описания прототипов всех пользовательских функций обработки прерываний и их регистрации.

```
void far BCIntNorm(unsigned wResult, unsigned dum1, unsigned dum2);
void far BCIntExc(unsigned wResult, unsigned wAW1, unsigned wAW2);
void far BCIntX(unsigned wResultX, unsigned wBase);
void far BCIntSig(unsigned wBase);
void far RTIntCmd(unsigned wCmd);
void far RTIntErr(unsigned wStatus);
```

```
void far RTIntData(unsigned wStatus);
void far MTIntX(unsigned wResultX, unsigned wBase);
void far MTIntSig(unsigned wBase);

bcdefintnorm(BCIntNorm);
bcdefintexc(BCIntExc);
bcdefintx(BCIntX);
bcdefintsig(BCIntSig);
rtdefintcmd(RTIntCmd);
rtdefinterr(RTIntErr);
rtdefintdata(RTIntData);
mtdefintx(MTIntX);
mtdefintsig(MTIntSig);
```

Для прерывания bcIntNorm в DOS добавлены два фиктивных параметра, что позволяет использовать общий обработчик с bcIntExc.

Определив причину прерывания, драйвер вызывает одну из заранее заданных пользовательских функций, соответствующих причине прерывания. При этом драйвер выполняет следующие действия. Сохраняется номер текущего выбранного устройства, выбирается в качестве текущего устройство, выдавшее прерывание, частично сохраняется состояние прервавшего устройства, формируются данные прерывания, соответствующие причине прерывания, после чего вызывается пользовательская функция. В этот момент прерывания самого процессора запрещены из-за того, что при любом прерывании сбрасывается в 0 флаг прерывания в слове состояния процессора. Драйвер не поддерживает вложенные прерывания (в том смысле, что возможно только однократное сохранение состояния, и вложенные прерывания от других устройств привели бы к потере сохраненных параметров для предыдущих прерываний). Пользовательская функция должна выполнять только минимально необходимую работу по регистрации прерывания и не должна разрешать прерывания процессора. После возврата из пользовательской функции в контроллер прерываний ПК выдается команда конца прерывания и осуществляется выход из прерывания.

Поскольку возможны ситуации, когда восстановление состояния устройств после прерывания не обязательно, вопрос о восстановлении должен решать сам программист. Для этого предназначены функции bcrestore, rtrestore и mtrestore, восстанавливающие сохраненное в начале прерывания состояние устройства КК, ОУ или МТ, вызвавшего прерывание, а также сохраненный номер выбранного устройства. При необходимости эти функции должны вызываться в конце пользовательской функции.

Если программисту недостаточно встроенных средств сохранения состояния при прерываниях (например, если надо восстанавливать состояние нескольких устройств), драйвер предоставляет возможность "ручного" программирования требуемых процедур (дополняющих стандартные). Для определения текущего состояния устройств предназначены функции: tmkselected, tmkgetmode, bcgetbase, mtgetbase, rtgetsubaddr, - что дает возможность в последующем восстановить измененные параметры.

Windows

В драйверах Windows работа с прерываниями осуществляется через механизм событий Win32. Прежде всего, процесс должен сообщить драйверу идентификатор (handle) используемого события для текущего выбранного ТМК через вызов функции tmkdefevent, которой передается в качестве первого параметра идентификатор события Win32, полученный из Win32 вызова CreateEvent:

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

Нулевое значение hEvent отменяет использование события. Значение параметра fEventSet равное TRUE указывает на необходимость установки события драйвером через вызовы SetEvent, а значение FALSE - через вызовы PulseEvent. Внимание! В текущей версии драйвера значение переменной fEventSet игнорируется (принимается равным TRUE). Кроме того, рекомендуется вызывать функцию ResetEvent непосредственно перед вызовом tmkdefevent.

Linux, QNX6

В драйвере tmk1553b работа с прерываниями осуществляется следующим образом. Введено понятие маски прерываний плат - это 32-разрядное число, в котором каждый из 31 младших бит соответствует прерыванию от платы с номером этого бита, т.е. бит 0 отвечает за прерывание ТМК 0, бит 1 - за прерывание ТМК 1 и т.д. Таким образом, в существующем виде драйвер может работать максимум с 31 устройством.

Для получения информации об возникших прерываниях или для ожидания прерываний процесс может вызвать функцию tmkwaitevents:

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

Параметр mask - это как раз маска прерываний, интересующих процесс. Если в момент вызова функции есть необработанные прерывания, подпадающие под указанную маску, то функция сразу возвращается и в качестве результата выдает маску существующих необработанных прерываний. Если же на момент вызова tmkwaitevents необработанных прерываний нет, то поведение функции зависит от параметра wait, который может принимать значения 0, -1 или положительное значение. При значении 0 происходит немедленный выход из функции (соответственно, маска ожидающих прерываний, возвращаемая функцией, будет равна 0, если ожидающих прерываний нет), а при значении -1 функция будет ждать возникновения хотя бы одного прерывания, подпадающего под маску. Если в параметре wait задано положительное значение, то функция воспринимает его как максимальное время ожидания в миллисекундах. В этом случае выход из функции производится либо по получению прерываний, либо по истечению времени ожидания. Отрицательный код возврата tmkwaitevents является кодом ошибки, произошедшей во время выполнения функции.

Linux, QNX6, Windows

После получения прерывания процесс может запросить структуру данных, описывающих прерывание, ставшее причиной события. Для этого предназначена функция tmkgetevd:

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

В качестве параметра передается указатель на структуру типа TTmkEventData, которую и заполняет драйвер соответствующими значениями:

```
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;
```

Память под структуру должна быть выделена заранее в самом процессе, драйвер только заполняет ее! Первыми двумя полями определяется тип прерывания и соответствующие этому типу дополнительные поля данных. Данные передаются те же самые, что и в DOS.

BcIntNorm	wMode=BC_MODE, nInt=1, данные bc.wResult
bcIntExc	wMode=BC_MODE, nInt=2, данные bc.wResult, bc.wAW1, bc.wAW2
bcIntX	wMode=BC_MODE, nInt=3, данные bcx.wResultX, bcx.wBase
bcIntSig	wMode=BC_MODE, nInt=4, данные bcx.wBase
rtIntCmd	wMode=RT_MODE, nInt=1, данные rt.wCmd
rtIntErr	wMode=RT_MODE, nInt=2, данные rt.wStatus
rtIntData	wMode=RT_MODE, nInt=3, данные rt.wStatus
mtIntX	wMode=MT_MODE, nInt=3, данные mt.wResultX, mt.wBase
mtIntSig	wMode=MT_MODE, nInt=4, данные mt.wBase
tmkIntOth	wMode=UNDEFINED_MODE, nInt=5, данные tmk.wRequest

Если возможна ситуация, когда до завершения обработки прерывания процессом могут возникнуть новые прерывания (а обычно это так и есть), то необходимо вызывать tmkgetevd в цикле до тех пор, пока в полученной структуре TTmkEventData поле номера прерывания не окажется нулевым (значения остальных полей структуры TTmkEventData в случае nInt=0 не определены). После этого снова можно ждать прерывания.

Также, надо учитывать, что новое прерывание может возникнуть еще во время цикла вызовов tmkgetevd для обработки предыдущих прерываний; тогда при обработке нового прерывания первый же вызов tmkgetevd даст нулевой номер прерывания, так как данные этого прерывания были прочитаны раньше.

Контроллер канала

В режиме КК ДОЗУ разбивается на базы - участки длиной 64 слова. Для задания обмена необходимо в какой-либо базе подготовить сообщение в соответствии с выбранным форматом (рис. 1), задать код управления для выбранного формата и запустить обмен из этой базы. После старта обмена в соответствии с командным словом (КС) (командными словами в режиме ОУ-ОУ) и кодом управления КК читает из ДОЗУ и передает в МК необходимые КС и информационные слова (ИС) и принимает из канала все передаваемые ОУ (обоими ОУ в режиме ОУ-ОУ) ответные слова (ОС) и ИС. Возможны два варианта завершения обмена: нормальное завершение и завершение с исключительной ситуацией. Нормальное завершение означает, что при обмене КК не обнаружил ошибок и в поле флагов ответных слов нет установленных битов. Завершение с исключительной ситуацией означает, что либо хотя бы в одном из принятых ОС есть установленные биты в поле флагов (что требует программной реакции), либо во время обмена в канале КК обнаружил ошибку. В зависимости от типа, ТМК аппаратно могут определять разный набор ошибок.

Все ТМК в режиме КК могут формировать следующие признаки:
- IB - в поле флагов хотя бы одного из принятых ОС установлен хотя бы один разряд;
- EBC - ошибка эхо контроля передачи (может возникать при неисправности КК, при неисправности ОУ - генерация в канале, при неисправности линии передачи информации (ЛПИ) - замыкание, при работе нескольких КК в МК);
- MEO - ошибка манчестерского кода при приеме (возникает при несоответствии принимаемых слов из МК стандарту - неправильный синхроимпульс, ошибка четности);
- TO - ошибка временных соотношений МК (возникает, когда временные интервалы между словами в сообщении не соответствуют стандартным - ошибка паузы до ответа ОУ и ошибка непрерывности ответа ОУ, а также при несоответствии числа передаваемых оконечным устройством ИС ожидаемому - ошибка числа слов, например при неготовности ОУ); более современные устройства (TMKX/TMKXI/TA/TAI) при этом позволяют различать ошибки паузы ОС (TOA) и ошибки паузы ИС (TOD);
- ERAO - ошибка адреса ОУ (возникает, когда значение в поле адреса ОС не совпадает с адресом ОУ, заданным в КС).

ТМК типов TMK400, RTMK400, TMKMPC могут формировать признак:
- EM - ошибка обмена с ДОЗУ (возникает, если в момент приема или передачи произошла недопустимая задержка предоставления доступа к ДОЗУ и слово было потеряно);

ТМК типов TMKX, TMKXI, TA, TAI могут формировать признаки:
- ELN - ошибка превышения числа слов (возникает, если ОУ отвечает числом слов большим, чем задано в команде);
- EBC - составная ошибка при приеме, совмещенная с ошибкой эхо контроля передачи (может возникать, если КК при приеме одновременно обнаружил несколько ошибок).

DOS

ТМК типов TMKX, TMKXI могут также формировать признаки:
- G1 - генерация в ЛПИ1 (дополнительный асинхронный бит - возникает в случае обнаружения непрерывной передачи в основной ЛПИ дольше 768 мкс и либо сбрасывается при прекращении генерации, либо может быть сброшен программным сбросом устройства);
- G2 - генерация в ЛПИ2 (дополнительный асинхронный бит - возникает в случае обнаружения непрерывной передачи в резервной ЛПИ дольше 768 мкс и либо

сбрасывается при прекращении генерации, либо может быть сброшен программным сбросом устройства);

Ни в одном драйвере (DOS, Linux, QNX6, Windows) нет поддержки обработки этих признаков. Поскольку такая обработка должна выполняться исключительно в обработчике прерывания, то существует возможность реализации пользовательской обработки только в драйвере DOS. Поэтому в Linux, QNX6 и в Windows безусловно, а также в DOS, если такая пользовательская обработка не реализована, программист не должен разрешать аппаратуре ТМК формировать данные признаки.

All

Во время обмена в МК возможна работа с другими базами КК - подготовка новых сообщений и обработка прежних.

Ввод устройства ТМК в режим КК с одновременным сбросом устройства, а при необходимости и повторная инициализация режима, осуществляется функцией bcreset.

Некоторые устройства (TMKX, TMKXI, TA, TAI) допускают настройку режима прерываний КК, в частности, блокировку прерываний по генерации в канале, а также отключение линии запроса прерывания устройства. Для управления такими устройствами могут использоваться константы:

- BC_GENER1_BL - режим блокировки прерываний по генерации в основной ЛПИ (только в TMKX, TMKXI);
- BC_GENER2_BL - режим блокировки прерываний по генерации в резервной ЛПИ (только в TMKX, TMKXI);
- TMK_IRQ_OFF - режим отключения линии запроса прерывания устройства.

Для настройки таких устройств в режиме КК предназначены функции bcdefirqmode и bcgetirqmode. Функция bcdefirqmode устанавливает режим, заданный комбинацией битов BC_GENER1_BL, BC_GENER2_BL и TMK_IRQ_OFF, для выбранной устройства и записывает его во внутреннюю переменную драйвера, а функция bcgetirqmode читает эту переменную. При вызове функции bcreset в эту переменную всегда записывается значение по умолчанию (BC_GENER1_BL|BC_GENER2_BL), задающее режим совместимости всех устройств.

Для работы с ДОЗУ программист должен сначала выбрать базу функцией bcdefbase. Функция bcgetmaxbase возвращает максимальный доступный номер базы на выбранном устройстве (номера баз лежат в диапазоне от 0 до этого максимального номера). Узнать номер выбранной базы можно функцией bcgetbase, основная область применения которой - пользовательские функции обработки прерываний. После выбора базы вся работа с ДОЗУ происходит именно в этой базе до выбора другой базы с помощью bcdefbase.

DOS

При прерываниях от КК или при вызове функции tmksave для КК номер текущей базы сохраняется драйвером, так что программист может работать в других базах, а при выходе из прерываний КК с вызовом bcrestore или при вызове tmkrestore восстанавливается в качестве текущей база с сохраненным номером.

All

В выбранной базе работают функции bcputw, bcgetw для записи, чтения слов в ДОЗУ и функции bcputblk, bcgetblk для записи, чтения блоков слов в ДОЗУ. С помощью этих функций в выбранной базе готовится сообщение для МК, и читаются результаты после завершения обмена.

Командные слова должны формироваться программистом самостоятельно и записываться с помощью этих же функций. Определены три макроса CW, CWM, CWMC, позволяющие сформировать командные слова из стандартных полей.

Макрос `CW(ADDR,DIR,SUBADDR,NWORDS)` предназначен для формирования командных слов приема/передачи данных. При этом можно пользоваться определенными константами направления передачи RT_RECEIVE (ОУ принимает) и RT_TRANSMIT (ОУ передает) для задания значения бита "Прием/передача" КС (DIR).

Макрос `CWM(ADDR,COMMAND)` предназначен для формирования командных слов режима управления. В качестве признака режима управления используется код 11111.

Макрос `CWMC(ADDR,0,COMMAND)` позволяет формировать командные слова режима управления с нулевым кодом признака режима управления.

Для задания кода команды можно использовать константы:

```
CMD_DYNAMIC_BUS_CONTROL
CMD_SYNCHRONIZE
CMD_TRANSMIT_STATUS_WORD
CMD_INITIATE_SELF_TEST
CMD_TRANSMITTER_SHUTDOWN
CMD_OVERRIDE_TRANSMITTER_SHUTDOWN
CMD_INHIBIT_TERMINAL_FLAG_BIT
CMD_OVERRIDE_INHIBIT_TERMINAL_FLAG_BIT
CMD_RESET_REMOTE_TERMINAL
CMD_TRANSMIT_VECTOR_WORD
CMD_SYNCHRONIZE_WITH_DATA_WORD
CMD_TRANSMIT_LAST_COMMAND_WORD
CMD_TRANSMIT_BUILT_IN_TEST_WORD
```

КС с групповым адресом должно в поле адреса содержать константу 31(дес.) = 11111(двоичн.).

При формировании команд передачи данных необходимо учитывать режим, в котором будет работать МК: с аппаратным битом или без аппаратного бита. В режиме без аппаратного бита в ОУ можно обращаться в 30 подадресов (от 1(дес.) = 00001(двоичн.) до 30(дес.) = 11110(двоичн.)), а для передачи команд режима управления значение поля подадреса/режима управления должно равняться 00000(двоичн.) или 11111(двоичн.). В режиме с аппаратным битом старший бит поля подадреса КС отводится под аппаратный бит, который всегда должен быть установлен в 1, число доступных подадресов сокращается до 15 (от 0(дес.) = 0000(двоичн.) до 14(дес.) = 1110(двоичн.), при этом в поле подадреса формируются значения от 10000(двоичн.) до 11110(двоичн.)), а для передачи команд режима управления значение поля подадреса/режима управления должно равняться 11111(двоичн.).

При формировании КС без использования макросов CW, CWM, CWMC необходимо помнить следующее.

В режиме с аппаратным битом можно пользоваться константами HBIT_MASK для установки в 1 аппаратного бита и CI_MASK - для установки режима управления 11111.

Значение поля длины блока данных в КС равное 00000(двоичн.) задает длину блока в 32 слова, а остальные значения 00001-11111(двоичн.) прямо соответствуют длине блока в 1-31 слово данных.

При формировании команд режима управления необходимо следовать табл. 2, следя за установкой в требуемое состояние бита "Прием/передача" КС.

При формировании команд режима управления надо обращать внимание на возможность применения групповой адресации. При этом каждая команда должна

применяться в соответствии со своим назначением. На многие стандартные команды устройства ТМК в режиме ОУ предусматривают определенную аппаратную реакцию. Команды управления "Принять управление каналом", "Синхронизация", "Синхронизация со словом данных", "Передать векторное слово" используются при нормальной работе системы, а остальные предназначены, в первую очередь, для обработки ошибок в МК.

Для режима КК существует два различных набора функций и констант для посылки сообщений в МК и обработки результатов. Первый набор, основанный на функциях bcdefbus/bcstart и стандартных кодах управления и кодах результатов обмена, работает на всех типах ТМК и обеспечивает однократный старт одного сообщения из указанной базы. Второй набор работает только на устройствах, поддерживающих выполнение цепочки сообщений (TMKX,TMKXI,TA,TAI), и позволяет задать однократный старт одного сообщения из указанной базы или цепочки сообщений, начиная с указанной базы. Этот набор основан на функциях bcdeflink/bcstartx/bcstop и расширенных кодах управления и кодах результатов обмена.

Сначала идет описание работы с первым набором.

В устройствах с резервированием возможен выбор одной из двух ЛПИ, основной или резервной, по которой будет работать КК. Для этого предназначена функция bcdefbus. Выбранная этой функцией линия будет использоваться для всех обменов до следующего вызова bcdefbus. Вызов этой функции для устройств без резервирования ЛПИ приводит к возникновению ошибки. Определены константы BUS_1 и BUS_A для основной линии, BUS_2 и BUS_B для резервной линии. С помощью функции bcgetbus можно узнать, какая линия выбрана в настоящий момент. Функция bcreset выбирает основную ЛПИ.

Сформированное в какой-либо базе сообщение можно запустить на обмен функцией bcstart. В качестве параметров функции передаются номер базы и код управления для устройства. После запуска обмена управление немедленно возвращается в программу, не дожидаясь завершения обмена. Функция bcstart не изменяет текущей выбранной базы для работы с ДОЗУ. Допустимые коды управления оформлены в виде констант:

- DATA_BC_RT - передача данных КК-ОУ (формат 1)
- DATA_BC_RT_BRCST - передача данных КК-ОУ в групповом режиме (формат 7)
- DATA_RT_BC - передача данных ОУ-КК (формат 2)
- DATA_RT_RT - передача данных ОУ-ОУ (формат 3)
- DATA_RT_RT_BRCST - передача данных ОУ-ОУ в групповом режиме (формат 8)
- CTRL_C_A - команда управления формата КС-ОС (формат 4)
- CTRL_C_BRCST - команда управления формата КС в групповом режиме (формат 9)
- CTRL_CD_A - команда управления формата КС+ИС-ОС (формат 6)
- CTRL_CD_BRCST - команда управления формата КС+ИС в групповом режиме (формат 10)
- CTRL_C_AD - команда управления формата КС-ОС+ИС (формат 5)

или

- CC_FMT_1 - передача данных КК-ОУ (формат 1)
- CC_FMT_2 - передача данных ОУ-КК (формат 2)
- CC_FMT_3 - передача данных ОУ-ОУ (формат 3)
- CC_FMT_4 - команда управления формата КС-ОС (формат 4)

- CC_FMT_5 - команда управления формата КС-ОС+ИС (формат 5)
- CC_FMT_6 - команда управления формата КС+ИС-ОС (формат 6)
- CC_FMT_7 - передача данных КК-ОУ в групповом режиме (формат 7)
- CC_FMT_8 - передача данных ОУ-ОУ в групповом режиме (формат 8)
- CC_FMT_9 - команда управления формата КС в групповом режиме (формат 9)
- CC_FMT_10 - команда управления формата КС+ИС в групповом режиме (формат 10)

После окончания выполнения сообщения, запущенного bcstart, КК выдает одно из двух прерываний завершения сообщения: bcIntNorm или bcIntExc. Для обоих прерываний драйвер формирует слово результата обмена, а при завершении с исключительной ситуацией также еще ответные слова, полученные КК. В сообщениях с одним ответным словом, второе ответное слово имеет неопределенное значение. При нормальном завершении слово результата всегда равно нулю. При завершении с исключительной ситуацией установленные биты в слове результата характеризуют причину такого завершения.

DOS

Чтобы была возможность использовать одну пользовательскую функцию завершения в обоих случаях, функции имеют одинаковые прототипы, однако при нормальном завершении параметры функции, соответствующие ответным словам, имеют неопределенное значение. Перед вызовом пользовательских функций завершения обмена драйвер устанавливает ту базу, из которой производился обмен с каналом.

All

Возможны различные варианты завершения с исключительной ситуацией: установлен только бит IB слова результата обмена, а биты ошибок сброшены, установлены бит IB слова результата обмена и хотя бы один бит ошибки, установлен хотя бы один бит ошибки в слове результата обмена, а бит IB сброшен. Константы, необходимые для анализа слова результата обмена:

DOS

- ERAO_MASK - маска бита ошибки адреса ОС;
- MEO_MASK - маска бита ошибки манчестерского кода при приеме;
- IB_MASK - маска бита, индицирующего наличие установленных разрядов в поле флагов ОС;
- TO_MASK - маска бита ошибки таймаута при приеме;
- EM_MASK - маска бита ошибки доступа к ДОЗУ со стороны канала (только для TMK400, RTMK400, TMKMPC);
- EBC_MASK - маска бита ошибки эхоконтроля при передаче;
- ELN_MASK - маска бита ошибки числа информационных слов (число ИС больше заданного) (только для TMKX/TMKXI/TA/TAI);
- G1_MASK - маска бита, индицирующего обнаружение приемником устройства непрерывной генерации в основной ЛПИ (только для TMKX/TMKXI);
- G2_MASK - маска бита, индицирующего обнаружение приемником устройства непрерывной генерации в резервной ЛПИ (только для TMKX/TMKXI);

Linux, QNX6, Windows

- S_ERAO_MASK - маска бита ошибки адреса ОС;
- S_MEO_MASK - маска бита ошибки манчестерского кода при приеме;
- S_IB_MASK - маска бита, индицирующего наличие установленных разрядов в поле флагов ОС;

- S_TO_MASK - маска бита ошибки таймаута при приеме;
- S_EM_MASK - маска бита ошибки доступа к ДОЗУ со стороны канала (только для TMK400, RTMK400, TMKMPC);
- S_EBC_MASK - маска бита ошибки эхоконтроля при передаче;
- S_ELN_MASK - маска бита ошибки числа информационных слов (число ИС больше заданного) (только для TMKX/TMKXI/TA/TAI);

All

Наиболее простой вариант - когда установлены только биты ошибок. В этом случае обмен считается недостоверным, ответные слова имеют неопределенное значение.

Если установлен только бит IB, то для определения реакции программы необходимо проанализировать ответное слово (два ответных слова в режиме ОУ-ОУ).

Константы, необходимые для анализа ответного слова:
- ADDRESS_MASK - маска адресной части ОС, адрес ответившего ОУ;
- ERROR_MASK - маска бита "Ошибка в сообщении", бит устанавливается, если хотя бы одно из ИС в полученном ОУ сообщении не удовлетворяет критериям достоверности или нарушены критерии достоверности сообщения, а также в случае, если ОУ получило запрещенную команду;
- HBIT_MASK - маска аппаратного бита, в ответном слове бит должен всегда быть нулевым;
- SREQ_MASK - маска бита "Запрос на обслуживание", бит устанавливается для сообщения активному КК о необходимости передачи или приема информации данным ОУ; если установка этого разряда может быть вызвана различными причинами, для определения содержания запроса КК запрашивает у ОУ векторное слово (команда "Передать векторное слово");
- NULL_MASK - маска битов, которые должны всегда быть нулевыми;
- BRCST_MASK - маска бита "Принята групповая команда", бит устанавливается, если достоверное КС предыдущего сообщения было с групповым адресом; поскольку форматы сообщений с групповым адресом предусматривают блокировку выдачи ОС, этот бит служит для определения того, что групповая команда была принята правильно;
- BUSY_MASK - маска бита "Подсистема занята", бит устанавливается для указания КК на то, что ОУ не может передавать данные в подсистему или получать данные из подсистемы в соответствии с командой КК;
- SSFL_MASK - маска бита "Неисправность подсистемы", бит устанавливается, если в подсистеме ОУ имеется неисправность, и данные, запрашиваемые у подсистемы, могут быть недостоверны; в случае, если ОУ может сопрягаться с несколькими подсистемами, сигналы признаков неисправности подсистем объединяются в этом бите, КК определяет неисправность через обычный обмен сообщениями, поскольку не существует команды режима управления, позволяющей получить слово встроенного контроля подсистемы;
- DNBA_MASK - маска бита "Принято управление каналом", бит устанавливается, если ОУ выполнил предложение активного КК принять управление каналом по команде "Динамическое управление";
- RTFL_MASK - маска бита "Неисправность терминала", бит устанавливается, если обнаружена неисправность в ОУ; КК имеет возможность блокировать или снять блокировку выдачи этого разряда посредством использования специальных команд режима управления; для получения более подробной информации о неисправности ОУ КК может использовать команду "Передать слово встроенного контроля".

Таким образом, установка разных битов в поле флагов ОС должна вызывать различную реакцию программы. Например, при установленном бите BUSY можно повторить сообщение сразу или через некоторое время, так как в этих случаях

сообщение недостоверно (при этом возможно также наличие признака ошибки ТО (TOD), если ОУ не передало все требуемые слова данных); при установленных битах SSFL и RTFL возможна недостоверность сообщения из-за сбоя на стороне ОУ; при установленном бите SREQ сообщение достоверно, но КК должен отреагировать на запрос обслуживания подсистемы ОУ; при установленных битах BRCST и DNBA сообщение достоверно в том случае, если существовали причины их установки, то есть либо в предыдущем сообщении была использована групповая адресация, либо КК пытался передать управление каналом ответившему битом DNBA ОУ, и наоборот, если в указанных случаях биты BRCST или DNBA не установлены, это указывает на какую-то ошибочную ситуацию.

Установленные одновременно бит IB и биты ошибок в слове результата обмена означают, что сначала КК принял достоверное ОС с установленными битами в поле флагов, а затем возникла та или иная ошибочная ситуация и обмен был прекращен. При этом программист может проанализировать ОС (в режиме ОУ-ОУ в этом случае, если в первом ОС есть установленные разряды, то второе ОС неопределено), но все сообщение считается недостоверным. Наиболее частая ситуация в нормально работающей системе - одновременная установка битов IB (из-за установленного бита BUSY ответного слова) и ТО, то есть подсистема ОУ занята, ОУ выдает ОС, но не выдает запрашиваемых данных.

Использование только уже описанных функций позволяет формировать и запускать отдельные сообщения одинаково на всех устройствах. Ниже описывается работа с функциями, которые позволяют запускать цепочки сообщений. Это второй набор функций запуска сообщений.

Основные отличия при работе с цепочками сообщений определяются алгоритмом работы КК в режиме цепочки сообщений. Такой режим задается установкой бита автоматического продолжения в расширенном коде управления при запуске сообщения.

Если бит автоматического продолжения не установлен, то КК после отработки сообщения вызывает прерывание и формирует слово результата обмена. Такой режим используется в функции bcstart, которая принудительно сбрасывает бит автоматического продолжения для запускаемого сообщения.

Для запуска цепочки сообщений введена функция bcstartx, позволяющая задать требуемое значение бита продолжения в расширенном коде управления. Для каждой базы в цепочке с помощью функции bcdeflink предварительно должны быть определены номер следующей базы цепочки и расширенный код управления следующей базы цепочки. Таким образом, если бит продолжения в коде управления установлен, то КК, в случае нормального завершения сообщения, _без_ выдачи прерывания читает ранее заданный код управления и номер базы для следующего сообщения в цепочке и запускает это сообщение. Если же обмен по МК завершился с исключительной ситуацией, или это было последнее сообщение цепочки, КК останавливается, вызывает прерывание и формирует слово результата обмена.

КК после вызовов bcstartx _не_ формирует прерываний bcIntNorm/bcIntExc. Вместо этого формируется одно прерывание bcIntX. Через параметры этого прерывания драйвер передает номер базы, из которой был произведен последний обмен, и слово результата обмена, которое имеет _отличающийся_ формат (с расширенной диагностикой ошибок) от формата слова результата при прерываниях bcIntNorm/bcIntExc.

DOS

При вызове пользовательской функции база, из которой производился последний обмен (и номер которой передается в качестве параметра), становится текущей выбранной базой.

All

Ответные слова отсутствуют в информации, передаваемой при этих прерываниях, но введена дополнительная функция bcgetansw, требующая в качестве параметра код формата сообщения и возвращающая двойное слово, младшие 16 разрядов которого содержат первое ответное слово в текущей выбранной базе (или значение FFFF, если заданный формат не предусматривает ответных слов), а старшие 16 разрядов - второе ответное слово (для формата ОУ-ОУ, или значение FFFF, для остальных форматов). Функция bcgetansw не может сама определить, были ли ответные слова получены на самом деле; она читает ячейки ДОЗУ, в которые должны записываться ответные слова в соответствии с текущим значением командного слова базы и заданным значением кода формата сообщения, поэтому решение о достоверности возвращенного этой функцией значения должно приниматься в соответствии с кодом результата обмена.

Цепочка сообщений может быть линейной (последнее сообщение имеет расширенный код управления со сброшенным битом продолжения), а может быть и циклической (в этом случае последнего сообщения, как такового, нет - все сообщения имеют установленный бит продолжения и ссылаются друг на друга по кругу; минимальным примером такого цикла может быть единственное сообщение, имеющее ссылку на себя). При выполнении цепочки сообщений КК не выдает никаких прерываний (прерывание формируется только при останове цепочки). Если необходимо синхронизировать выполнение программы с передачей конкретных сообщений цепочки, можно установить в расширенном коде управления для нужных баз специальный бит сигнального прерывания. Если в момент _загрузки_ в КК кода управления (т.е. непосредственно перед стартом из базы с таким кодом управления) в этом коде установлен бит сигнального прерывания, КК вызывает дополнительное сигнальное прерывание bcIntSig. В качестве параметра этого прерывания драйвер передает номер базы, из которой в момент вызова производился обмен с МК.

DOS

Текущая выбранная база при этом не изменяется, хотя и сохраняется во внутренней переменной драйвера и может быть стандартно восстановлена вызовом bcrestore при выходе из прерывания. В отличие от остальных прерываний при выходе из сигнального прерывания автоматически восстанавливается номер текущего выбранного устройства, поэтому вызывать bcrestore необходимо только, если в пользовательской функции переопределялась текущая база, и нужно восстановить ее номер, бывший в момент прерывания.

All

Необходимо отметить, что в системах с резервированием ЛПИ вызов функции bcdefbus задает ЛПИ только для обменов, запускаемых функцией bcstart. В сообщениях, запускаемых функцией bcstartx, ЛПИ выбирается отдельно для каждой базы соответствующим битом в расширенном коде управления.

Запуск цепочки сообщений осуществляется функцией bcstartx, имеющей формат вызова аналогичный формату bcstart, но допускающей задание расширенных кодов управления. Расширенный код управления состоит из:

- собственно кода управления CX_CC_MASK, определяющего формат сообщения (константы кода управления форматом полностью совпадают с кодами управления для функции bcstart);
- бита CX_BUS_MASK задания ЛПИ (значения CX_BUS_0, CX_BUS_A (или 0) выбирают основную ЛПИ, а CX_BUS_1, CX_BUS_B - резервную ЛПИ);
- бита CX_CONT_MASK автоматического продолжения (значение CX_STOP (или 0) задает останов цепочки после отработки этого сообщения, а значение CX_CONT задает автоматический запуск следующего сообщения в цепочке после отработки этого сообщения);
- бита CX_SIG_MASK разрешения выдачи сигнального прерывания (значение CX_NOSIG (или 0) запрещает выдачу сигнального прерывания, а значение CX_SIG разрешает выдачу сигнального прерывания в момент _загрузки_(!) кода управления в КК).

Для формирования цепочки сообщений предназначена функция bcdeflink, имеющая формат вызова аналогичный формату bcstartx, и задающая для текущей выбранной базы следующую базу в цепочке и расширенный код управления для сообщения в следующей базе цепочки. Установленные bcdeflink значения можно получить, вызвав функцию bcgetlink, возвращающую двойное слово, младшие 16 разрядов которого содержат номер следующей базы в цепочке, а старшие 16 разрядов - расширенный код управления для следующей базы в цепочке, который можно проанализировать с помощью констант выделения полей CX_CC_MASK, CX_BUS_MASK, CX_CONT_MASK, CX_SIG_MASK. Функции bcdeflink и bcgetlink работают только в текущей выбранной базе, то есть для задания всей цепочки необходимо для каждой базы в цепочке (начиная с первой и кончая предпоследней) выполнить пару вызовов bcdefbase - bcdeflink. Для последней базы в линейной цепочке, выполнять вызов bcdeflink не требуется.

При необходимости можно прервать цепочку сообщений, вызвав функцию bcstop. Эта функция не имеет параметров, а ее вызов приводит к обнулению бита автоматического продолжения в сообщении, которое в этот момент передается в канал. В результате КК после окончания отработки этого сообщения останавливается и, как обычно, формирует прерывание bcIntX.

Слово расширенного результата обмена, формируемое КК после вызовов bcstartx состоит из нескольких полей, которые можно выделить и проанализировать с помощью констант:

- SX_ERR_MASK - маска поля кода ошибки, содержащего свернутый код ошибки от 0 до 7 (0 означает отсутствие ошибок);
- SX_IB_MASK - маска бита, индицирующего наличие установленных разрядов в поле флагов ОС;
- SX_G1_MASK - маска бита, индицирующего обнаружение приемником устройства непрерывной генерации в основной ЛПИ (только TMKX/TMKXI);
- SX_G2_MASK - маска бита, индицирующего обнаружение приемником устройства непрерывной генерации в резервной ЛПИ (только TMKX/TMKXI).

Считается, что нормальному завершению сообщения соответствует код результата с нулевыми значениями полей SX_ERROR и SX_IB. Если поле SX_ERROR содержит ненулевой код ошибки и/или установлен бит SX_IB, то обмен считается завершенным с исключительной ситуацией. Константы поля кода ошибок:

- SX_NOERR - 0 - обмен завершен без ошибок;
- SX_MEO - 1 - ошибка четности или ошибка манчестерского кода;

- SX_TOA - 2 - неверная пауза перед ответным словом (отсутствует ответное слово);
- SX_TOD - 3 - нарушена непрерывность сообщения (отсутствует слово (слова) данных);
- SX_ELN - 4 - число информационных слов больше заданного;
- SX_ERAO - 5 - неверный адрес ОУ в ОС;
- SX_ESYN - 6 - ошибка типа синхроимпульса;
- SX_EBC - 7 - ошибка эхоконтроля при передаче или комбинация нескольких ошибок при приеме.

Между кодами ошибки в слове результата обмена bcstartx и битами ошибок в слове результата обмена bcstart установлено такое соответствие: ошибки SX_MEO, SX_ESYN приводят к установке бита MEO_MASK (S_MEO_MASK); ошибки SX_TOA, SX_TOD - к установке бита TO_MASK (S_TO_MASK); ошибка SX_ELN - к установке бита ELN_MASK (S_ELN_MASK); ошибка SX_ERAO - к установке бита ERAO_MASK (S_ERAO_MASK); ошибка SX_EBC - к установке бита EBC_MASK (S_EBC_MASK).

Кроме задания сигнальных прерываний существует еще один способ слежения за базой, с которой работает в данный момент КК. Функция bcgetstate возвращает двойное слово, младшие 16 разрядов которого содержат номер базы, с которой происходит работа, а старшие 16 разрядов - текущее слово состояния. Вызов этой функции приводит к сбросу необслуженного сигнального прерывания, если оно в этот момент было, поэтому совместно использовать функцию bcgetstate и возможность задания сигнальных прерываний следует с осторожностью.

Монитор

Режим монитора во многом сходен по принципам управления и форматам функций с режимом контроллера канала. Ввод устройства ТМК в режим МТ с одновременным сбросом устройства, а при необходимости и повторная инициализация режима, осуществляется функцией mtreset. Необходимо проверять возвращаемое функцией значение, так как устройство может не поддерживать режим монитора.

Устройства, поддерживающие режим монитора (TMKX, TMKXI, TA, TAI), допускают настройку режима прерываний МТ, в частности, блокировку прерываний по генерации в канале, а также отключение линии запроса прерывания устройства. Для управления такими устройствами могут использоваться константы:
- MT_GENER1_BL - режим блокировки прерываний по генерации в основной ЛПИ (только в TMKX, TMKXI);
- MT_GENER2_BL - режим блокировки прерываний по генерации в резервной ЛПИ (только в TMKX, TMKXI);
- TMK_IRQ_OFF - режим отключения линии запроса прерывания устройства.

Для настройки таких устройств в режиме МТ предназначены функции mtdefirqmode и mtgetirqmode. Функция mtdefirqmode устанавливает режим, заданный комбинацией битов MT_GENER1_BL, MT_GENER2_BL и TMK_IRQ_OFF, для выбранного устройства и записывает его во внутреннюю переменную драйвера, а функция mtgetirqmode читает эту переменную. При вызове функции mtreset в эту переменную всегда записывается значение по умолчанию (MT_GENER1_BL|MT_GENER2_BL).

В режиме МТ ДОЗУ разбивается на базы - участки длиной 64 слова. Для работы с ДОЗУ программист должен сначала выбрать базу функцией mtdefbase. Функция mtgetmaxbase возвращает максимальный доступный номер базы на выбранном устройстве (номера баз лежат в диапазоне от 0 до этого максимального номера). Узнать номер выбранной базы можно функцией mtgetbase, основная область применения

которой - пользовательские функции обработки прерываний. После выбора базы вся работа с ДОЗУ происходит именно в этой базе до выбора другой базы с помощью mtdefbase.

DOS

При прерываниях от МТ или при вызове функции tmksave для МТ номер текущей базы сохраняется драйвером, так что программист может работать в других базах, а при выходе из прерываний МТ с вызовом mtrestore или при вызове tmkrestore восстанавливается в качестве текущей база с сохраненным номером.

All

В выбранной базе работают функции mtputw, mtgetw для записи, чтения слов в ДОЗУ и функции mtputblk, mtgetblk для записи, чтения блоков слов в ДОЗУ. При обычной работе используются только функции чтения из ДОЗУ mtgetw и mtgetblk, с помощью которых получается информация об отслеженных монитором сообщениях в МК. Функции записи могут использоваться, например, для тестирования ДОЗУ.

При работе монитор формирует два типа прерываний: mtIntX и mtIntSig. Прерывание mtIntX вызывается при останове монитора. Через параметры этого прерывания драйвер передает номер базы, в которую было записано последнее сообщение, и слово состояния сообщения.

DOS

Эта же база становится активной. Для восстановления номера текущей базы перед выходом из пользовательской функции надо вызвать функцию mtrestore.

All

При выполнении цепочки МТ не выдает прерываний (прерывание формируется только при останове монитора). Если необходимо синхронизировать выполнение программы с работой монитора, можно установить в коде управления для нужных баз специальный бит сигнального прерывания. Если в момент загрузки в МТ кода управления (т.е. непосредственно перед стартом из базы с таким кодом управления) в этом коде установлен бит сигнального прерывания, МТ вызывает дополнительное сигнальное прерывание mtIntSig. В качестве параметра этого прерывания драйвер передает номер текущей стартовой базы.

DOS

Текущая выбранная база при вызове пользовательской функции не изменяется, хотя и сохраняется во внутренней переменной драйвера и может быть стандартно восстановлена вызовом mtrestore при выходе из прерывания. В отличие от остальных прерываний при выходе из сигнального прерывания автоматически восстанавливается номер текущей выбранной устройства, поэтому вызывать mtrestore необходимо только, если в пользовательской функции переопределялась текущая база, и нужно восстановить ее номер, бывший в момент прерывания.

All

Для запуска монитора необходимо сформировать в базах с помощью вызовов функции mtdeflink программу монитора, описывающую последовательность используемых баз, аналогично КК в режиме цепочек сообщений, а затем запустить монитор функцией mtstartx, передав ей в качестве параметров номер первой базы в цепочке и расширенный код управления для нее. Функция mtdeflink задает для текущей выбранной базы следующую базу в цепочке и расширенный код управления для следующей базы цепочки. В режиме МТ расширенный код управления включает в себя:

- бит CX_CONT_MASK автоматического продолжения (значение CX_STOP (или 0) задает останов цепочки после отработки этой базы, а значение CX_CONT задает автоматический переход к следующей базе в цепочке после отработки этой базы);
- бит CX_INT_MASK запрета прерывания работы монитора при обнаружении исключительной ситуации - установленных битов в ОС или ошибок в сообщении в МК в этой базе (значение CX_INT разрешает останов монитора по исключительной ситуации, а CX_NOINT - вынуждает монитор перейти к следующей базе в цепочке, даже если в сообщении была обнаружена исключительная ситуация, естественно, только если установлен бит автоматического продолжения);
- бит CX_SIG_MASK разрешения выдачи сигнального прерывания (значение CX_NOSIG (или 0) запрещает выдачу сигнального прерывания, а значение CX_SIG разрешает выдачу сигнального прерывания в момент _загрузки_(!) кода управления в МТ).

Установленные mtdeflink значения можно получить, вызвав функцию mtgetlink, возвращающую двойное слово, младшие 16 разрядов которого содержат номер следующей базы в цепочке, а старшие 16 разрядов - расширенный код управления для следующей базы в цепочке, который можно проанализировать с помощью констант выделения полей CX_CONT_MASK, CX_INT_MASK, CX_SIG_MASK. Функции mtdeflink и mtgetlink работают только в текущей выбранной базе, то есть для задания всей цепочки необходимо для каждой базы в цепочке (начиная с первой и кончая предпоследней) выполнить пару вызовов mtdefbase - mtdeflink. Для последней базы в линейной цепочке, выполнять вызов mtdeflink не требуется.

После запуска функцией mtstartx, монитор входит в активный режим прослушивания МК. В этом режиме монитор пытается распознать все сообщения, выдаваемые в канал. Управление при этом немедленно возвращается в программу. Функция mtstartx не изменяет текущей выбранной базы для работы с ДОЗУ. Распознав сообщение, монитор записывает все слова (командные, ответные и информационные) в порядке их поступления в ту базу, которая была указана при старте монитора. После приема сообщения, монитор формирует на это сообщение слово состояния. Нормальное завершение означает, что при обмене МТ не обнаружил ошибок, и в поле флагов ответных слов нет установленных битов. Завершение с исключительной ситуацией означает, что либо хотя бы в одном из принятых ОС есть установленные биты в поле флагов (что требует программной реакции), либо во время обмена в канале МТ обнаружил ошибку. Далее монитор может либо продолжить работу, либо остановиться. Продолжит работу монитор в том случае, если в текущем коде управления был установлен бит автоматического продолжения CX_CONT и, либо в сообщении не было обнаружено ничего, приводящего к исключительной ситуации (нет ни ошибок, ни установленных битов в ОС), либо возникла исключительная ситуация, но был установлен бит запрета прерывания работы монитора CX_NOINT в текущем коде управления). Если монитор продолжает работу, то он записывает слово состояния сообщения в текущую стартовую базу, читает номер следующей стартовой базы в цепочке и расширенный код управления для нее, в этот момент, если в этом коде управления установлен бит сигнального прерывания, монитор вырабатывает прерывание mtIntSig, а затем снова переходит в активный режим прослушивания МК уже с новой базой в качестве стартовой. В качестве параметра прерывания mtIntSig передается номер текущей стартовой базы. Останавливается монитор, если в текущем коде управления был сброшен бит автоматического продолжения CX_STOP, или возникла исключительная ситуация при сброшенном бите запрета прерывания работы монитора CX_INT. При останове монитор не записывает слово состояния сообщения в базу. При этом вырабатывается прерывание mtIntX. В качестве параметров прерывания mtIntX передаются номер последней стартовой базы и слово состояния сообщения, записанного в эту базу. Далее

программа может в соответствии с кодом формата сообщения, битами состояния и кодом ошибки, содержащимся в слове состояния прочитать с помощью вызовов mtgetw, mtgetblk все слова (КС, ОС и ИС) сообщения.

DOS

При вызове пользовательской функции прерывания mtIntX база, в которую было записано последнее сообщение, становится текущей выбранной базой.

All

Чтобы прочитать слово состояния сообщения для текущей выбранной базы в том случае, если на этом сообщении монитор не остановился и записал слово состояния в базу, предназначена функция mtgetsw.

Формат слова состояния приведен на рис. 5. Оно состоит из нескольких полей, которые можно выделить и проанализировать с помощью констант:
- SX_BUS_MASK - маска бита, показывающего, по какой из линий передачи принято сообщение в системах с резервированием ЛПИ (SX_BUS_0, SX_BUS_A - основная ЛПИ, SX_BUS_1, SX_BUS_B - резервная ЛПИ);
- SX_ME_MASK - маска бита интегрированного признака ошибки, этот бит может быть установлен даже в том случае, если поле кода ошибки не содержит никакого кода ошибки, т.е. ошибка не подпадает ни под одну из характеристик, оговоренных для этого поля, но тем не менее монитор признал сообщение ошибочным;
- SX_SCC_MASK - маска поля кода формата сообщения (см.ниже);
- SX_K1_MASK - маска бита, указывающего, что МТ выявил ошибку в первом КС;
- SX_K2_MASK - маска бита, указывающего, что МТ выявил ошибку во втором КС;
- SX_G1_MASK - маска бита, индицирующего обнаружение приемником устройства непрерывной генерации в основной ЛПИ (только в TMKX, TMKXI);
- SX_G2_MASK - маска бита, индицирующего обнаружение приемником устройства непрерывной генерации в резервной ЛПИ (только в TMKX, TMKXI);
- SX_IB_MASK - маска бита, индицирующего наличие установленных разрядов в поле флагов ОС;
- SX_ERR_MASK - маска поля кода ошибки, содержащего свернутый код ошибки от 0 до 7 (0 означает отсутствие ошибок).

Сообщение считается нормальным только в том случае, если равны нулю и бит интегрированного признака ошибки в сообщении (SX_ME_MASK) и поле кода ошибки (SX_ERR_MASK), так как возможны ситуации, когда код ошибки равен нулю (SX_NOERR), но бит интегрированного признака ошибки все же установлен. Если поля ERR или ME содержат ненулевой код ошибки и/или установлен бит IB, то обмен считается завершенным с исключительной ситуацией. Константы, необходимые для анализа слова результата обмена:

- SX_NOERR - 0 - обмен завершен без ошибок;
- SX_MEO - 1 - ошибка четности или ошибка манчестерского кода;
- SX_TOA - 2 - неверная пауза перед ответным словом (отсутствует ответное слово);
- SX_TOD - 3 - нарушена непрерывность сообщения (отсутствует слово (слова) данных);
- SX_ELN - 4 - число информационных слов больше заданного;
- SX_ERAO - 5 - неверный адрес ОУ в ОС;
- SX_ESYN - 6 - ошибка типа синхроимпульса;
- SX_EBC - 7 - комбинация нескольких ошибок (в TMKX, TMKXI) или вытеснение (в TA, TAI).

Код формата сообщения можно выделить с помощью константы SX_SCC_MASK. Если его затем сдвинуть на 10 разрядов вправо, получится код, совпадающий с кодами управления форматом сообщений для КК, которые оформлены в виде констант:

- DATA_BC_RT - передача данных КК-ОУ (формат 1)
- DATA_BC_RT_BRCST - передача данных КК-ОУ в групповом режиме (формат 7)
- DATA_RT_BC - передача данных ОУ-КК (формат 2)
- DATA_RT_RT - передача данных ОУ-ОУ (формат 3)
- DATA_RT_RT_BRCST - передача данных ОУ-ОУ в групповом режиме (формат 8)
- CTRL_C_A - команда управления формата КС-ОС (формат 4)
- CTRL_C_BRCST - команда управления формата КС в групповом режиме (формат 9)
- CTRL_CD_A - команда управления формата КС+ИС-ОС (формат 6)
- CTRL_CD_BRCST - команда управления формата КС+ИС в групповом режиме (формат 10)
- CTRL_C_AD - команда управления формата КС-ОС+ИС (формат 5)

или

- CC_FMT_1 - передача данных КК-ОУ (формат 1)
- CC_FMT_2 - передача данных ОУ-КК (формат 2)
- CC_FMT_3 - передача данных ОУ-ОУ (формат 3)
- CC_FMT_4 - команда управления формата КС-ОС (формат 4)
- CC_FMT_5 - команда управления формата КС-ОС+ИС (формат 5)
- CC_FMT_6 - команда управления формата КС+ИС-ОС (формат 6)
- CC_FMT_7 - передача данных КК-ОУ в групповом режиме (формат 7)
- CC_FMT_8 - передача данных ОУ-ОУ в групповом режиме (формат 8)
- CC_FMT_9 - команда управления формата КС в групповом режиме (формат 9)
- CC_FMT_10 - команда управления формата КС+ИС в групповом режиме (формат 10)

При необходимости можно прервать цепочку сообщений, вызвав функцию mtstop. Эта функция не имеет параметров, а ее вызов приводит к обнулению бита автоматического продолжения для текущей стартовой базы. В результате МТ после окончания отработки текущего сообщения останавливается и, как обычно, формирует прерывание mtIntX. Но, если в момент вызова mtstop в линии нет обмена, то МТ так и останется в состоянии ожидания окончания текущего обмена (который еще не начался). Если ждать окончания этого обмена не требуется (например, программа должна завершить работу), то для немедленного останова монитора можно использовать функцию mtreset.

Кроме задания сигнальных прерываний существует еще один способ слежения за базой, с которой работает в данный момент МТ. Функция mtgetstate возвращает двойное слово, младшие 16 разрядов которого содержат номер базы, с которой происходит работа, а старшие 16 разрядов - текущее слово состояния, в котором при необходимости можно проанализировать состояние битов, индицирующих обнаружение МТ генерации в канале. Вызов этой функции приводит к сбросу необслуженного сигнального прерывания, если оно в этот момент было, поэтому совместно использовать функцию mtgetstate и возможность задания сигнальных прерываний следует с осторожностью.

Оконечное устройство

В режиме ОУ ДОЗУ разбивается на подадреса - участки длиной 32 слова. Таким образом, каждому подадресу в ОУ, адресуемому контроллером, соответствует свой участок памяти. При этом различаются подадреса на прием и подадреса на передачу. В одном ОУ доступно максимум 30 подадресов (в режиме без аппаратного бита, а в режиме с аппаратным битом - 15 подадресов). Области памяти, соответствующие неиспользуемым подадресам (00000 и 11111(двоичн.)), в разных типах устройств могут служить для хранения флаговых слов (в режиме ОУ с флагами) и ИС для команд режима управления с присоединенным ИС. Требуемый объем памяти составляет 2 * (30 + 2) * 32 = 2048 слов. Если объем памяти, установленной на устройстве, превышает 2К слов (например, 8К = 8192 слов), то вся память делится на несколько страниц объемом по 2К слов, которые можно программно переключать. При этом устройства либо допускают раздельное задание страницы, доступной со стороны МК, и страницы, доступной со стороны ПК (устройства TMKX), либо предусматривают задание одной и той же рабочей страницы как для МК, так и для ПК (устройства TMK400, RTMK400).

В режиме ОУ устройство вырабатывает как минимум два типа прерывания: прерывание по приходу команды режима управления rtIntCmd и прерывание по ошибке rtIntErr. Прерывание по приходу команды режима управления вырабатывается не для всех команд, а только для тех, которые не могут быть выполнены устройством самостоятельно. Прерывание по ошибке вырабатывается в том случае, если устройство приняло КС, адресуемое данному ОУ, и обнаружена ошибка формата сообщения, ошибка манчестерского кода или ошибка четности. Прочитав при этом слово состояния устройства, можно определить разряды последнего КС, при обработке которого возникла ошибка. Конкретная причина ошибки не определяется.

По умолчанию, ОУ работают в режиме, в котором команды приема/передачи данных никаких прерываний не вызывают. Старые платы (TMK400, RTMK400, TMKMPC) не имели возможности формировать такие прерывания вообще. В более новых платах (TMKX, TMKXI, TA, TAI) есть возможность включения режима формирования прерываний по приему/передаче данных rtIntData. Если прерывания rtIntData не используются, то для определения факта приема/передачи данных ОУ необходимо производить программный опрос устройства, вид которого зависит от режима работы устройства. Если обращения к конкретному ОУ достаточно редки, можно рекомендовать либо опрос состояния ОУ по таймеру, либо сопровождение передач данных дополнительной командой режима управления, которая вызовет прерывание в ОУ.

DOS

В качестве таймера можно использовать либо системный таймер с частотой 18.2 Гц, либо таймер, входящий в подсистему Real Time Clock IBM PC AT, функции поддержки которого включены в библиотеку tmkml.c.

All

Режимы работы ОУ либо определяются распайкой или установкой перемычек на устройстве (для устройств TMK400, TMKMPC, TMKX, TMKXI), либо должны задаваться программно (для устройств RTMK400, TMKX, TMKXI, TA, TAI). Поведение ОУ определяется комбинацией из трех режимов, которые могут быть раздельно включены или выключены:

- режим использования аппаратного бита (бит режима RT_HBIT_MODE);
- режим работы с флагами (бит режима RT_FLAG_MODE);
- режим приема сообщений с групповым адресом (бит режима RT_BRCST_MODE).

На устройствах TMKX, TMKXI установкой перемычек задается только режим использования аппаратного бита.

На устройствах, не допускающих программного задания режимов работы ОУ, изготовителем устанавливаются режимы использования аппаратного бита и приема сообщений с групповым адресом. При необходимости изменить режимы работы ОУ, в зависимости от конкретного типа устройства, может потребоваться либо переустановить, либо перепаять перемычки.

Для устройств, допускающих программное задание режимов работы ОУ, предназначены функции rtdefmode и rtgetmode. Функция rtdefmode устанавливает режим, заданный комбинацией битов RT_HBIT_MODE, RT_FLAG_MODE и RT_BRCST_MODE, для выбранного устройства и записывает его во внутреннюю переменную драйвера, а функция rtgetmode читает эту переменную. При вызове функции tmkconfig в эту переменную всегда записывается значение по умолчанию (RT_HBIT_MODE|RT_BRCST_MODE), а вызов функции rtreset приводит к установке режимов работы ОУ в соответствии со значением этой переменной.

Кроме того, некоторые устройства (TMKX, TMKXI, TA, TAI) допускают настройку режима прерываний ОУ, в частности, блокировку прерываний по генерации в канале, блокировку прерываний по приему/передаче данных, а также отключение линии запроса прерывания устройства. Для управления такими устройствами предназначены константы:
- RT_GENER1_BL - режим блокировки прерываний по генерации в основной ЛПИ (только в TMKX, TMKXI);
- RT_GENER2_BL - режим блокировки прерываний по генерации в резервной ЛПИ (только в TMKX, TMKXI);
- RT_DATA_BL - режим блокировки прерываний по приему/передаче данных;
- TMK_IRQ_OFF - режим отключения линии запроса прерывания устройства.

Для настройки таких устройств в режиме ОУ предназначены функции rtdefirqmode и rtgetirqmode. Функция rtdefirqmode устанавливает режим, заданный комбинацией битов RT_GENER1_BL, RT_GENER2_BL, RT_DATA_BL, TMK_IRQ_OFF, для выбранного устройства и записывает его во внутреннюю переменную драйвера, а функция rtgetirqmode читает эту переменную. При вызове функции rtreset в эту переменную всегда записывается значение по умолчанию (RT_GENER1_BL|RT_GENER2_BL|RT_DATA_BL), задающее режим совместимости всех устройств.

Режим использования аппаратного бита определяет, будет ли ОУ реагировать на командные слова только с установленным аппаратным битом или на любые командные слова (в этом случае аппаратный бит расширяет поле подадреса ОУ и может принимать любые значения). Режим использования или неиспользования аппаратного бита должен быть установлен одинаково для всех ОУ в МК! Включение режима используется для повышения надежности работы канала за счет введения различий между командными и ответными словами, имеющими одинаковый синхроимпульс (аппаратный бит должен быть установлен в командном слове и сброшен в ответном), и для облегчения работы монитора канала. Выключение режима обеспечивает большую гибкость адресации за счет расширения поля подадреса ОУ в командном слове.

Режим приема сообщений с групповым адресом определяет, будет ли ОУ реагировать на командные слова с групповым адресом (11111 двоичн.). Если режим включен, ОУ реагирует как на сообщения с адресом этого ОУ, так и на групповые сообщения. При выключенном режиме ОУ реагирует только на сообщения, адресованные непосредственно ему. Для разных ОУ в одном канале можно независимо включать или выключать этот режим в зависимости от системных требований.

Режим использования флагов определяет способ арбитража доступа к подадресам в ДОЗУ между каналом и ПК. При этом выбирается один из двух возможных режимов: режим с флагами или режим без флагов. Более простым является режим без флагов; он может применяться в системах, где допустима потеря данных или повторное использование данных (например, КК считывает периодически состояние какого-либо датчика, или устанавливает состояние какого-либо регулятора, при этом КК с какой-то периодичностью обращается к ОУ, а процессор ПК ОУ в цикле читает или обновляет данные в ДОЗУ), или точно заранее известно расписание передач по МК. В этом режиме по любой команде КК принять или передать данные ОУ принимает данные и записывает их в указанный подадрес или читает данные из указанного подадреса и передает их в МК. Режим ОУ с флагами должен использоваться в системах, где недопустима потеря или повторное использование данных, или неизвестно заранее расписание передач по МК. В этом режиме для каждого подадреса (подадреса на прием и на передачу раздельны) существует так называемое флаговое слово, определяющее готовность данных в соответствующем подадресе. Флаговые слова расположены в ДОЗУ в области неиспользуемых подадресов.

Ввод устройства ТМК в режим ОУ с одновременным сбросом устройства, а при необходимости и повторная инициализация режима, осуществляется функцией rtreset. При этом устанавливается нулевая рабочая страница в ДОЗУ.

Для задания адреса ОУ в МК предназначена функция rtdefaddress. Для определения установленного адреса предназначена функция rtgetaddress, при этом, если устройство позволяет определение установленного адреса (например, TMK400, RTMK400, TA, TAI), то адрес читается из устройства, а если такая возможность отсутствует, то адрес читается из внутренней переменной драйвера.

На всех платах, кроме TMK400 и TMKMPC, ОУ может быть включено и выключено функцией rtenable. ОУ включается вызовом rtenable(RT_ENABLE), а выключается вызовом rtenable(RT_DISABLE). Получить текущее состояние (RT_ENABLE или RT_DISABLE) можно через вызов rtenable(RT_GET_ENABLE).

Установлен следующий алгоритм работы функций rtreset и rtdefaddress для всех ОУ. Если rtreset вызывается первый раз после tmkconfig или bcreset или mtreset, то устройство переводится в режим ОУ, но остается выключенным как после вызова rtenable(RT_DISABLE). Включается устройство только после задания его адреса через rtdefaddress. Повторные вызовы rtreset не меняют состояние включено/выключено.

Для выбора рабочей страницы в ДОЗУ предназначена функция rtdefpage. Если устройство позволяет раздельно задавать страницу для МК и страницу для ПК, можно пользоваться функциями rtdefpagebus, rtdefpagepc, но обращение к этим функциям для устройств, не имеющих такой возможности, будет вызывать ошибочную ситуацию. Функция rtgetmaxpage возвращает максимальный доступный номер страницы на выбранном устройстве (номера страниц лежат в диапазоне от 0 до этого максимального номера). Узнать номер выбранной страницы можно функцией rtgetpage (номера раздельных страниц для МК и ПК возвращаются функциями rtgetpagebus и rtgetpagepc). Переключать страницы во время работы необходимо с осторожностью, чтобы не возникла ситуация переключения страницы во время обмена с МК, то есть момент переключения должен быть синхронизирован с работой КК. В большинстве случаев можно (и, наверное, желательно) обойтись работой в одной странице.

Для работы с ДОЗУ программист должен сначала выбрать подадрес функцией rtdefsubaddr. В пределах одной страницы возможен доступ к 30 подадресам приема и 30

подадресам передачи. Используемый диапазон подадресов зависит от режима работы МК - с аппаратным битом (10000 - 11110(двоичн.)) или без аппаратного бита (00001 - 11110(двоичн.)). Конкретный режим задается распайкой или установкой перемычек на устройстве или программируется функцией rtdefmode и влияет только на функционирование ОУ (в режиме с аппаратным битом ОУ не отвечает на команды с аппаратным битом равным 0). Для выбора подадреса приема или передачи можно пользоваться константами RT_RECEIVE (ОУ принимает) и RT_TRANSMIT (ОУ передает). Функция rtdefsubaddr не проверяет заданные подадрес и направление передачи, а корректирует их по умолчанию, обнуляя "лишние" биты, поэтому программист должен сам следить за тем, чтобы не задавались "неправильные" подадреса 00000(двоичн.) и 11111(двоичн.). Узнать выбранный подадрес можно функцией rtgetsubaddr, основная область применения которой - пользовательские функции обработки прерываний. Эта функция возвращает одно слово, младшие пять бит которого содержат текущий подадрес, а направление передачи подадреса задается битом "прием/передача", который эквивалентен такому же биту КС (так же расположен в слове и принимает те же значения). Для выделения этого бита можно пользоваться константой RT_DIR_MASK, а полученное значение будет равно RT_RECEIVE для подадреса приема и RT_TRANSMIT для подадреса передачи. После выбора подадреса вся работа с ДОЗУ происходит именно в этом подадресе до выбора другого подадреса с помощью rtdefsubaddr.

DOS

При вызове функции tmksave для ОУ текущий подадрес сохраняется драйвером, так что программист может работать в других подадресах, а при вызове tmkrestore восстанавливается в качестве текущего сохраненный подадрес.

All

В выбранном подадресе работают функции rtputw, rtgetw для записи, чтения слов в ДОЗУ и функции rtputblk, rtgetblk для записи, чтения блоков слов в ДОЗУ. С помощью этих функций в выбранном подадресе передачи программист готовит всю информацию для передачи в МК, а из выбранного подадреса приема получает информацию, принятую из МК.

В режиме без флагов программа может в любой момент обращаться к любому подадресу, с которым в этот момент не ведет обмен МК. Возможны два алгоритма работы без флагов, обеспечивающих разрешение конфликтов при доступе в один тот же подадрес МК и ПК. В соответствии с одним программа должна определить занятость выбранного (функцией rtdefsubaddr) подадреса с помощью функции rtbusy, возвращающей нулевое значение, если выбранный подадрес свободен, и ненулевое, если выбранный подадрес занят. Под занятым подразумевается подадрес, с которым в данный момент МК ведет обмен (при этом различаются подадреса приема и передачи). Если подадрес занят, то необходимо ждать его освобождения, снова вызывая rtbusy, а если свободен, то через время не более 16 мкс (при чтении) или 20 мкс (при записи) после вызова rtbusy программа обязана прочитать или записать слово данных и далее читать или писать их с циклом не более 20 мкс. Другой алгоритм предусматривает блокировку подадреса функцией rtlock. Эта функция имеет параметры, как у функции rtdefsubaddr, и также оставляет выбранным указанный ей подадрес. Однако, после вызова rtlock, МК не будет иметь доступа к выбранному подадресу, и на все обращения к этому подадресу ОУ будет отвечать ОС с установленным битом BUSY "Подсистема занята", а программа может работать с блокированным подадресом в любом темпе в течение любого времени. Исключение представляет тот случай, когда rtlock вызывается в момент, когда МК ведет уже ведет обмен с блокируемым подадресом. В этой ситуации обмен не прерывается, а блокировка подадреса задерживается до момента окончания обмена. Для обнаружения этой ситуации необходимо использовать функцию rtbusy,

которая будет сообщать о занятости подадреса до окончания обмена. Когда подадрес блокирован, rtbusy всегда возвращает нулевое значение. Для разблокировки блокированного подадреса могут использоваться функции rtunlock, rtlock, rtdefsubaddr. После функции rtunlock текущий выбранный подадрес остается прежним, но ни один подадрес не блокирован. Функция rtlock разблокирует текущий подадрес, устанавливает новый подадрес и блокирует его. Функция rtdefsubaddr разблокирует текущий подадрес и устанавливает новый подадрес, не блокируя его. В каждый момент времени может быть заблокирован только один подадрес.

Алгоритм работы с флагами следующий. Передающая сторона перед записью данных в ДОЗУ должна убедиться, что флаг выбранного подадреса сброшен, только в этом случае записать свои данные и установить флаг подадреса. Принимающая сторона перед чтением данных из ДОЗУ должна убедиться, что флаг выбранного подадреса установлен, только в этом случае прочитать свои данные и сбросить флаг подадреса. Таким образом, при передаче данных из ОУ устройство проверяет флаговое слово адресуемого подадреса передачи, и, если флаг установлен (новые данные в подадресе готовы), читает данные из ДОЗУ и передает их в МК, а если флаг сброшен (новые данные в подадресе не готовы, чтение из подадреса со стороны МК запрещено), отвечает ОС с установленным битом BUSY "Подсистема занята", данные из ДОЗУ не читает и не передает в МК. При приеме данных в ОУ устройство проверяет флаговое слово адресуемого подадреса приема, и если флаг сброшен (прежние данные в подадресе прочитаны ПК), принимает данные из МК и записывает их в ДОЗУ, а если флаг установлен (прежние данные в подадресе еще не прочитаны в ПК, запись в подадрес со стороны МК запрещена), принимает данные из МК, проверяя их на ошибки, но не записывает их в ДОЗУ, а в конце отвечает ОС с установленным битом BUSY "Подсистема занята". При обращении к ДОЗУ со стороны ПК программа должна сама читать, проверять и записывать флаги в соответствии с описанным алгоритмом для всех подадресов, с которыми собирается работать: устанавливать после записи информации в подадрес передачи для разрешения передачи и ждать сброса, сбрасывать для разрешения приема в подадрес приема и ждать установки для чтения информации из подадреса. Для этого предназначены функции rtsetflag, rtclrflag, rtgetflag. Функция rtgetflag имеет параметры, как у функции rtdefsubaddr, читает флаговое слово для заданного подадреса и устанавливает этот подадрес в качестве текущего выбранного подадреса. Функции rtsetflag и rtclrflag устанавливают и сбрасывают флаг для выбранного ранее подадреса (функциями rtdefsubaddr или rtgetflag). Кроме этих функций в драйвер включены функции rtgetflags и rtputflags, позволяющие прочитать и записать сразу несколько флаговых слов и работающие с буферным массивом в ОЗУ ПК. После вызова этих функций текущий выбранный подадрес не определен, и необходимо его задание, например, функцией rtdefsubaddr. Формат флагового слова приведен на рис.4.

Программист имеет возможность управлять значением отдельных бит в ОС, для чего предназначены функции rtsetanswbits и rtclranswbits, позволяющие устанавливать и сбрасывать определенные биты ОС. С помощью функции rtgetanswbits можно узнать текущее состояние этих бит. Для выбора бит можно пользоваться константами:

- SREQ - бит "Запрос подсистемы на обслуживание" устанавливается, если подсистема, подключенная к данному ОУ, требует какого-либо обслуживания со стороны КК; если причин установки этого бита может быть несколько, конкретная причина должна быть указана в векторном слове, которое КК может получить по команде "Передать векторное слово";

- BUSY - бит "Подсистема занята" устанавливается, если в подсистеме производятся какие-то действия, во время которых обмен с МК нежелателен (например, тестирование

ДОЗУ). При установке этого бита устройство не выполняет команд КК на прием/передачу данных, отвечая только ОС с установленным битом BUSY; если бит программно не устанавливался, устройство может его устанавливать самостоятельно для отдельных команд обмена данными в соответствии с описанными выше алгоритмами работы;

- SSFL - бит "Неисправность подсистемы" устанавливается, если в подсистеме, подключенной к данному ОУ, обнаружена какая-то неисправность, что говорит КК о возможной недостоверности данных при обмене с этим ОУ; бит может устанавливаться и сбрасываться только программно;

- RTFL - бит "Неисправность терминала" устанавливается, если обнаружена какая-то неисправность в самой устройстве ТМК или в другом оборудовании, считающемся в данной системе частью терминала МК, устройство не имеет встроенных средств самоконтроля, поэтому бит может устанавливаться и сбрасываться только программно; КК имеет возможность аппаратно маскировать выдачу этого бита в ОС с помощью команд режима управления "Подавить бит неисправности терминала" и "Отменить подавление бита неисправности терминала"; конкретный тип неисправности может быть указан в слове встроенного контроля, которое КК может получить по команде "Передать слово встроенного контроля";

- DNBA - бит "Готовность принятия управления каналом" устанавливается ОУ, если оно в состоянии в любой момент принять управление каналом на себя (стать очередным активным КК) по команде режима управления "Динамическое управление каналом"; если бит программой не установлен, то на указанную команду ОУ отвечает ОС со сброшенным битом "Принято управление каналом", то есть предложение КК принять управление не выполнено; если бит предварительно установлен программой, то на указанную команду ОУ отвечает ОС с установленным битом "Принято управление каналом" и продолжает отвечать такими ОС на все следующие команды КК (до команды "Динамическое управление" бит в ОС не устанавливается даже при установленном программой бите DNBA), при этом программный сброс бита DNBA не приводит к прекращению выдачи ОС с установленным битом DNBA; прекратить выдачу таких ОС можно либо вызвав функцию rtreset, либо из КК командой режима управления "Установить исходное состояние ОУ".

Для установки или сброса сразу нескольких бит при вызове функций rtsetanswbits и rtclranswbits можно объединить несколько указанных констант по "ИЛИ". Установка или сброс выбранных бит не влияет на установленное ранее состояние остальных бит.

Чтобы задать или прочитать ИС для команд режима управления с присоединенным ИС ("Передать векторное слово", "Передать слово встроенного контроля", "Синхронизация со словом данных", "Блокировать выбранный передатчик", "Разблокировать выбранный передатчик") предназначены функции драйвера rtputcmddata и rtgetcmddata. Как параметры им задается команда, для которой необходимо задать или прочитать ИС и само ИС для функции rtputcmddata, функция rtgetcmddata возвращает прочитанное слово в качестве результата. Задаваемая команда должна содержать сам код команды (младшие пять бит) и бит "прием/передача", аналогичный биту "прием/передача" командного слова. Можно использовать константы кодов команд:

```
CMD_DYNAMIC_BUS_CONTROL
CMD_SYNCHRONIZE
CMD_TRANSMIT_STATUS_WORD
CMD_INITIATE_SELF_TEST
CMD_TRANSMITTER_SHUTDOWN
CMD_OVERRIDE_TRANSMITTER_SHUTDOWN
CMD_INHIBIT_TERMINAL_FLAG_BIT
```

```
CMD_OVERRIDE_INHIBIT_TERMINAL_FLAG_BIT
CMD_RESET_REMOTE_TERMINAL
CMD_TRANSMIT_VECTOR_WORD
CMD_SYNCHRONIZE_WITH_DATA_WORD
CMD_TRANSMIT_LAST_COMMAND_WORD
CMD_TRANSMIT_BUILT_IN_TEST_WORD
```

Для определения текущего состояния ОУ можно пользоваться функцией rtgetstate, которая возвращает слово состояния (рис.3). Младшие 11 бит слова состояния отображают соответствующие биты последнего КС, обработанного (или обрабатываемого в данный момент) ОУ. Кроме того, слово состояния содержит бит занятости подадреса, бит "Ошибка при обмене с подсистемой" (только в старых платах TMK400, RTMK400, TMKMPC), бит "Ошибка при обмене с МК". Бит занятости подадреса устанавливается в бесфлаговом режиме, если подадрес, в котором происходит работа программы в данный момент, совпадает с подадресом, адресуемом соответствующими битами КС. Появление установленного бита "Ошибка при обмене с подсистемой" может указывать на неисправность устройства ТМК. Бит "Ошибка при обмене с МК" устанавливается, если ОУ обнаружил в адресуемом ему сообщении ошибки формата или кода "Манчестер-2".

Как уже было сказано, ОУ формирует прерывания rtIntCmd, rtIntErr и rtIntData. Для прерывания rtIntCmd драйвер передает в качестве параметра код пришедшей команды, включающий 5 младших бит собственно команды и бит "прием/передача" КС. Для прерываний rtIntErr драйвер передает в качестве параметра слово состояния ОУ с установленным битом ошибки в МК, а по остальным битам слова состояния можно определить команду, при выполнении которой обнаружена ошибка. Так как в устройствах TMK400, RTMK400, TMKMPC, TMKX, TMKXI нет буферизации прерываний, при непрерывном поступлении с минимально паузой команд, адресованных одному и тому же ОУ, возможна ситуация, когда информация о состоянии прерывания доступна для чтения из устройства менее, чем в течение 20 мкс. Если система не успевает за это время вызвать обработчик прерывания драйвера, возможно появление неоднозначности при обработке прерывания. Обычно это может быть пропуск прерывания, кроме того, на платах TMK400, RTMK400, TMKMPC возможно формирование прерывания rtIntErr без установленного бита ошибки (биты слова состояния в этом случае содержат биты нового КС).

DOS

Перед вызовом пользовательских функций драйвер сохраняет номер выбранного устройства и текущий подадрес устройства, вызвавшего прерывание. При необходимости восстановить в конце прерывания состояние устройства и драйвера, должна быть вызвана функция rtrestore.

Многоадресное ОУ

Linux, QNX6, Windows

Начиная с версии v2.00 драйвер поддерживает работу с платами TX6/TE6 (MRTX, MRTXI) в режиме многоадресного ОУ (МОУ), а начиная с версии v4.00 также поддерживается работа с платами TA1-PCI-32RT (MRTAI). В этом режиме на одном физическом устройстве многоадресного ОУ реализуются 4 виртуальных ОУ (ВОУ) с разными адресами в канале на платах MRTX, MRTXI или 31 ВОУ с разными адресами в канале на платах MRTAI. Если задать при конфигурации драйвера для какого-либо устройства режим МОУ, в списке устройств драйвера будет как это устройство с номером, заданным в реестре (конфигурации), так и дополнительно несколько ВОУ с номерами, присвоенными самим драйвером.

Для работы с МОУ введены функции:

```
TMK_DATA rtenable(TMK_DATA rtEnable);
int mrtgetmaxn();
DWORD mrtconfig(int mrtNumber);
int mrtselected();
TMK_DATA mrtgetstate();
void mrtdefbrcsubaddr0();
int mrtdefbrcpage(TMK_DATA mrtBrcPage);
TMK_DATA mrtgetbrcpage();
int mrtreset();
```

Основное назначение функции rtenable - включение или выключение работы текущего ВОУ в МОУ. Однако с некоторыми ограничениями этой функцией можно пользоваться и для обычных ОУ. Исключение составляют ОУ на платах TMK400 и TMKMPC, которые не могут быть выключены этой функцией. ОУ включается вызовом rtenable(RT_ENABLE), а выключается вызовом rtenable(RT_DISABLE). Получить текущее состояние (RT_ENABLE или RT_DISABLE) можно через вызов rtenable(RT_GET_ENABLE).

Алгоритм работы функций rtreset и rtdefaddress для ВОУ такой же, как и обычных ОУ. Если rtreset вызывается первый раз после tmkconfig или bcreset или mtreset, то устройство переводится в режим ОУ, но остается выключенным как после вызова rtenable(RT_DISABLE). Включается устройство только после задания его адреса через rtdefaddress. Повторные вызовы rtreset не меняют состояние включено/выключено.

Функция mrtgetmaxn возвращает максимальный возможный номер МОУ. Этот номер равен максимальному возможному номеру физического устройства в драйвере. То есть для драйвера, предназначенного для работы с 8 устройствами (с 0 по 7) mrtgetmaxn возвратит 7. Функция tmkgetmaxn без сконфигурированных в реестре (конфигурации) МОУ работает также. При наличии МОУ драйвер присвоит входящим в них ВОУ номера, начиная с 8, а tmkgetmaxn будет возвращать максимальный присвоенный номер ВОУ. То есть, если будет сконфигурировано одно МОУ с 4 ВОУ, то tmkgetmaxn вернет 11.

Функция mrtconfig может вызываться вместо tmkconfig для МОУ, если программа пишется специально для работы с МОУ. Эта функция конфигурирует целиком МОУ и все ВОУ, входящие в него, для работы с процессом, вызвавшим mrtconfig. Функция mrtconfig возвращает двойное слово, младшие 16 бит которого содержат начальный номер ВОУ, принадлежащих этому МОУ, а старшие 16 бит - число ВОУ, принадлежащих этому МОУ. При ошибке (например, если заданный номер не является номером МОУ) функция вернет нулевое значение.

Возможна работа с многоадресными ОУ двумя способами. Можно захватить само МОУ и все входящие в него ВОУ вызовом mrtconfig. Другой способ - не вызывать mrtconfig для МОУ, а вызывать tmkconfig для каждого конкретного ВОУ. Необходимо помнить, что МОУ может иметь некоторые общие для всех входящих ВОУ ресурсы, например, арбитр доступа к подадресу БЗУ (используется в функциях rtlock, rtunlock, rtbusy), биты управления режимами работы (функции rtdefmode, rtgetmode, rtdefirqmode, rtgetirqmode), регистр аппаратного сброса (функции rtreset, mrtreset). Если разные ВОУ захвачены разными процессами через tmkconfig, то вызов этих функций для одного ВОУ, может влиять на работу остальных. Функция rtreset, если она вызвана для ВОУ, выполнит полный аппаратный сброс МОУ только в том случае, если захвачено только это ВОУ, а

МОУ и остальные ВОУ не захвачены никаким процессом. Функция mrtreset выполнит аппаратный сброс МОУ и всех входящих в него ВОУ в любом случае.

Захваченные через mrtconfig или tmkconfig МОУ и/или ВОУ выбираются для работы, как и ранее, функцией tmkselect.

Функция mrtselected позволяет узнать номер выбранного МОУ для текущего выбранного ВОУ. Для обычных ОУ функции tmkselected и mrtselected будут возвращать одно и то же значение. Для ВОУ tmkselected будет возвращать номер ВОУ, а mrtselected - номер МОУ.

Функция mrtreset выполняет аппаратный сброс и инициализацию МОУ и всех входящих ВОУ.

Для ВОУ и МОУ не могут вызываться функции bcreset и mtreset, поэтому в универсальных программах, если они могут использоваться на компьютерах с установленными МОУ, желательно проверять код возврата из этих функций. Для обычных устройств код возврата bcreset будет нулевым. Для МОУ/ВОУ bcreset вернет ненулевой код ошибки.

Функция mrtgetstate позволяет прочитать регистр слова состояния МОУ (см. техническое описание режима МОУ). Функция rtgetstate, используемая для чтения регистра слова состояния обычных ОУ, для ВОУ прочитает виртуальное слово состояния, формируемое самим драйвером.

По умолчанию, драйвер копирует все приходящие в МОУ сообщения с групповым адресом во все ВОУ. Однако, можно работать и отдельно с областью памяти, выделенной в МОУ для групповых сообщений. В MRTX, MRTXI - это отдельный подадрес памяти, а в MRTA, MRTAI - это отдельная страница памяти.

Функция mrtdefbrcsubaddr0 (только в MRTX, MRTXI) позволяет задать в качестве текущего подадреса МОУ или ВОУ с младшим номером подадрес, в который приходят все групповые сообщения (см. техническое описание режима МОУ MRTX, MRTXI). Необходимо помнить, что даже если ВОУ с младшим номером выключено, но включено какое либо другое в составе данного МОУ, то в этот подадрес будут поступать данные от сообщений с групповым адресом.

Функция mrtdefbrcpage (только в MRTA, MRTAI) позволяет выбрать в качестве текущей страницы ВОУ страницу групповых сообщений (аппаратно являющуюся ВОУ с адресом 31). Хотя функция имеет параметр - номер страницы, в данной версии драйвера можно указывать только 0 в качестве параметра. Функция mrtgetbrcpage позволяет узнать текущий номер страницы групповых сообщений, который также всегда 0 в данной версии. Функция rtgetpage позволяет определить, является ли текущая выбранная страница обычной страницей или страницей для групповых сообщений. Если результат rtgetpage меньше или равен результату rtgetmaxpage, то выбрана обычная страница. Иначе результат rtgetpage больше, чем результат rtgetmaxpage, и фактически он равен rtgetmaxpage() + 1 + mrtgetbrcpage().

ПРИЛОЖЕНИЕ
Рисунки и таблицы

При описании полей форматов слов в скобках даны определенные константы, которыми можно пользоваться для выделения описываемых полей.

Форматы сообщений с указанием
размещения сообщения МК в базе ДОЗУ КК
при длине массива 32 слова.

Смещение в базе:	00	01	02		…		32	33	34	35		…		63
КК–ОУ (ф.1)	КС	ИС	ИС	…	…	ИС	ИС	ОС						
ОУ–КК (ф.2)	КС	ОС	ИС	ИС	…	…	ИС	ИС						
ОУ–ОУ (ф.3)	КС1	КС2	ОС2	ИС	ИС	…	…	ИС	ИС	ОС1				
КС–ОС (ф.4)	КС	ОС												
КС–ОС+ИС (ф.5)	КС	ОС	ИС											
КС+ИС–ОС (ф.6)	КС	ИС	ОС											
КК–ОУгр.(ф.7)	КС	ИС	ИС	…	…	ИС	ИС							
ОУ–ОУгр.(ф.8)	КС1	КС2	ОС2	ИС	ИС	…	…	ИС	ИС					
КСгр.(ф.9)	КС													
КС+ИСгр.(ф.10)	КС	ИС												

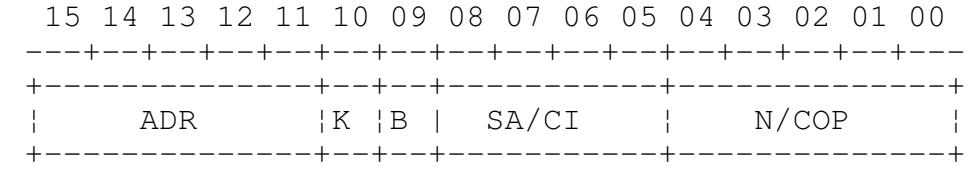
Форматы КК-ОУ, ОУ-КК, ОУ-ОУ, КК-ОУгр., ОУ-ОУгр. являются форматами передачи данных, а форматы КС-ОС, КС-ОС+ИС, КС+ИС-ОС, КСгр., КС+ИСгр. - форматами режима управления.

ПРИМЕЧАНИЕ. ИС в режимах ОУ-КК, ОУ-ОУ, КС-ОС+ИС и ОС во всех режимах записываются в ДОЗУ после запуска обмена.

Рис.1.

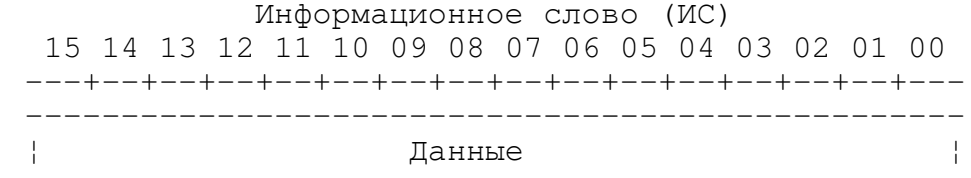
ФОРМАТЫ СЛОВ МК.

Командное слово (КС)

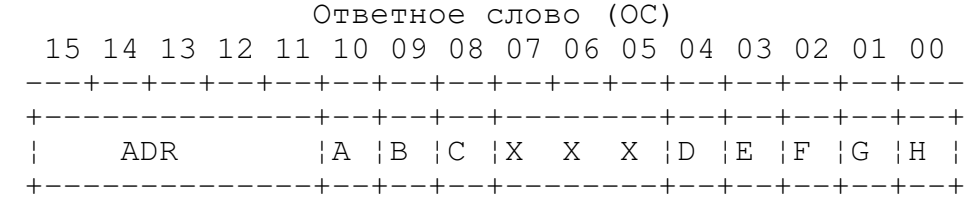


ADR - поле адреса адресуемого ОУ:
00000-11110 - адрес ОУ;
11111(31) - групповой адрес;
К - бит "прием/передача", указывает на действия ОУ:
0 - ОУ принимает информацию (RT_RECEIVE),
1 - ОУ передает информацию (RT_TRANSMIT);
В - аппаратный бит:
0/1 - старший бит подадреса в режиме без аппаратного бита,
1 - в режиме с аппаратным битом (HBIT_MASK);
SA/CI - подадрес/режим управления (вместе с битом В):
00000, 11111 - задает команду режима управления в поле N/COP в режиме без аппаратного бита,
11111 - задает команду режима управления в поле N/COP в режиме с аппаратным битом (CI_MASK),
00001-11110 - задает подадрес в режиме без аппаратного бита,
10000-11110 - задает подадрес в режиме с аппаратным битом;
N/COP - число информационных слов/код команды:
00000 - 32 слова в режиме передачи данных,
00001-11111 - 1-31 слово в режиме передачи данных,
00000-11111 - код команды в режиме управления.

Информационное слово (ИС)



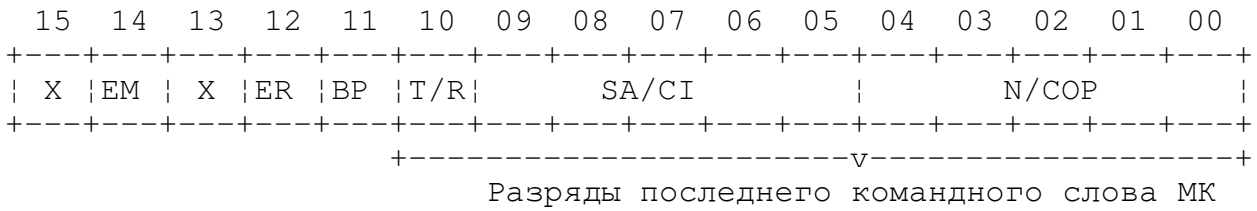
Ответное слово (ОС)



ADR - поле адреса ОУ ответившего ОУ:
00000-11110 - адрес ОУ (ADDR_MASK);
А - бит ошибки сообщения (ERROR_MASK);
В - аппаратный бит (HBIT_MASK);
С - бит "Запрос обслуживания подсистемы" (SREQ_MASK);
XXX - резерв (000) (NULL_MASK);
D - бит "Принята групповая команда" (BRCST_MASK);
E - бит "Подсистема занята" (BUSY_MASK);
F - бит "Неисправность подсистемы" (SSFL_MASK);
G - бит "Принято управление каналом" (DNBA_MASK);
H - бит "Неисправность терминала" (RTFL_MASK);

Рис.2.

Формат слова состояния ОУ.

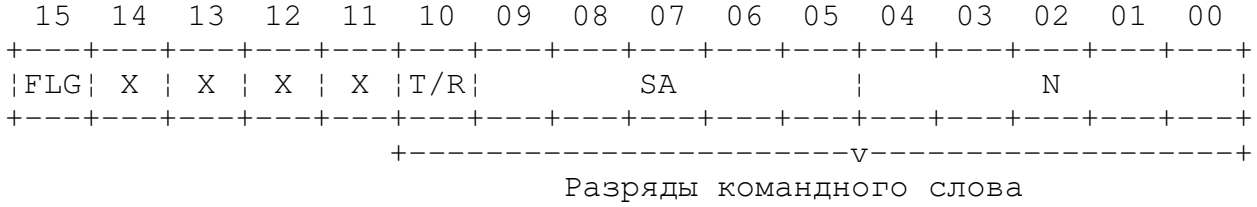


ПРИМЕЧАНИЕ: слово состояния ОУ возвращается функцией rtgetstate, а также передается в качестве параметра прерываний rtIntErr, rtIntData.

T/R - разряд "прием/передача" последнего КС (RT_DIR_MASK);
SA/CI - разряды подадрес/режим управления последнего КС (SUBADDR_MASK, CI_MASK);
N/COP - разряды число слов/код команды последнего КС (NWORS_MASK, CMD_MASK);
BP - бит занятости выбранного подадреса в режиме без флагов устанавливается, если выбранный подадрес ведет в данный момент обмен с МК;
ER - бит "Ошибка обмена с подсистемой", установка этого бита может указывать на неисправность в устройстве (этот бит может устанавливаться только в платах TMK400, RTMK400, TMKMPC);
EM - бит "Ошибка сообщения МК" устанавливается, если в адресуемом данному ОУ сообщении обнаружена ошибка формата сообщения, ошибка манчестерского кода или ошибка четности (RT_ERROR_MASK).

Рис.3.

Формат флагового слова ОУ.

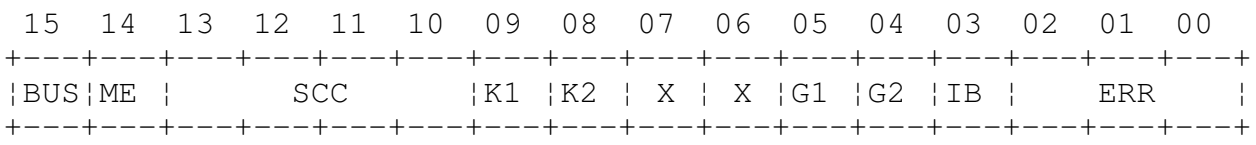


ПРИМЕЧАНИЕ: при записи флагового слова в ДОЗУ с помощью функции rtputflags необходимо задавать только сам бит флага FLG, а при анализе прочитанных функциями rtgetflag, rtgetflags флаговых слов можно использовать и поля T/R, SA, N, которые заполняются устройством при изменении флагового слова при обмене с МК. Например, можно узнать конкретное число слов, принятое/переданное в/из данного подадреса. Функция rtsetflag записывает в флаговое слово константу 8000(шестн.), а функция rtclrflag - константу 0000.

T/R - разряд "прием/передача" КС (RT_DIR_MASK);
SA/CI - разряды подадреса КС (SUBADDR_MASK);
N/COP - разряды числа слов КС (NWORS_MASK);
FLG - бит флага готовности данных (RT_FLAG, RT_FLAG_MASK).

Рис.4.

Формат слова состояния сообщения (MT).



ПРИМЕЧАНИЕ: слово состояния сообщения MT возвращается функцией mtgetsw для текущей выбранной базы, а также передается в качестве параметра прерывания mtIntX для последнего принятого сообщения (слово состояния, передаваемое при останове монитора, в дальнейшем невозможно получить функцией mtgetsw).

BUS - бит номера ЛПИ, по которой принято сообщение (SX_BUS_MASK);
ME - бит интегрированного признака ошибки в сообщении (SX_ME_MASK);
SCC - поле кода формата сообщения, при его сдвиге на 10 разрядов вправо совпадает с кодом управления КК (SX_SCC_MASK);
K1 - бит признака ошибки в первом КС (SX_K1_MASK);
K2 - бит признака ошибки во втором КС (SX_K2_MASK);
G1 - бит признака обнаружения генерации в основной ЛПИ ТМКХ (SX_G1_MASK);
G2 - бит признака обнаружения генерации в резервной ЛПИ ТМКХ (SX_G2_MASK);
IB - бит признака установленных битов в поле флагов любого ОС (SX_IB_MASK);
ERR - поле кода ошибки сообщения (SX_ERR_MASK).

Рис.5.

Кодировка поля SA/CI в командах МК. Таблица 1.

Код SA/CI	Подадрес или режим управления
00000,11111	Признак записи команды в поле числа слов. При остальных кодах в поле числа слов записана размерность пересылаемого массива данных от одного(00001) до тридцати двух(00000).
00001 ... 11110	Подадрес. Определяется адрес вводимых или выдаваемых данных внутри терминала.

Кодировка поля N/COP в командах МК. Таблица 2.

Код N/COP	Функции	Бит "прием/передача"	Присоединение ИС	Возможность применения групповой команды
	Команды режима управления формата КС–ОС			
00000	Динамическое управление	1	–	–
00001	Синхронизация	1	–	+
00010	Передать ответное слово	1	–	–
00011	Начать самоконтроль	1	–	+
00100	Блокировать передатчик	1	–	+
00101	Разблокировать передатчик	1	–	+
00110	Подавить бит флага терминала	1	–	+
00111	Отменить подавление бита флага терминала	1	–	+
01000	Установить исходное состояние ОУ	1	–	+
01001 . . . 01111	Резерв Резерв	1 1	– –	– –
	Команды режима управления форматов КС+ИС–ОС и КС–ОС+ИС			
10000	Передать векторное слово	1	+	–
10001	Синхронизация с ИС	0	+	+
10010	Передать последнее командное слово	1	+	–
10011	Передать слово встроенного контроля	1	+	–
10100	Блокировать выбранный передатчик	0	+	+
10101	Разблокировать выбранный передатчик	0	+	+
10110 . . . 11111	Резерв Резерв	X X	+ +	– –

ПРИМЕЧАНИЕ. X – произвольное значение бита.

При формировании команд в КК и их обработке в ОУ необходимо учитывать состояние бита "прием/передача". Драйвер всегда обрабатывает команды как слова состоящие из самого кода команды (младшие 5 бит) и бита "прием/передача". Ниже приведено описание команд и особенностей их реализации на устройствах ТМК.

Команды "Синхронизация" и "Синхронизация со словом данных" предназначены для осуществления синхронизации работы КК и ОУ.

Команда "Передать векторное слово" связана с битом запроса на обслуживание в ОС и используется для определения вида обслуживания, если у терминала имеется более одного вида запроса на обслуживание. Программа ОУ должна предварительно задать векторное слово через функцию rtputcmddata.

Команда "Принять управление каналом" ("Динамическое управление") предлагает ОУ стать новым КК, приняв управление каналом. Если ОУ в состоянии это сделать, то оно должно ответить ОС с установленным битом "Принято управление каналом". Программа ОУ, способная перевести устройство в режим КК и далее управлять каналом, должна предварительно установить функцией rtsetanswbits(DNBA) бит "Готов принять управление каналом". После получения указанной команды такое ОУ станет устанавливать в каждом ОС бит "Принято управление каналом". Если по какой-либо причине необходимо продолжить работу в режиме ОУ, то для прекращения выдачи установленного бита в ОС программа ОУ должна инициализировать ОУ функцией rtreset. Для отмены действия команды самим КК можно использовать команду "Установить исходное состояние ОУ".

Команда "Установить исходное состояние ОУ" может использоваться КК для инициализации выбранного ОУ. Данная команда выполняется устройством аппаратно. На устройствах TMK400, RTMK400, TMKMPC, TMKX, TMKXI команда не вызывает прерывания, а на устройствах TA, TAI команда вызывает прерывание в ПК. В многоадресных ОУ и возможных будущих реализациях ОУ данная команда может вызывать прерывание.

Команда "Передать последнее командное слово" позволяет определить последнее достоверное КС, принятую и выполненную ОУ до данной команды. Она не изменяет битов ответного слова ОУ, что позволяет использовать ее в операциях по обнаружению и исправлению ошибок, не влияя на ОС, что могло бы внести новые ошибки. Данная команда выполняется устройством аппаратно и не вызывает прерывания в ПК.

Команда "Передать ответное слово" также не изменяет битов ОС и позволяет получить контроллеру обновленные данные о прохождении информации по МК и об оборудовании, влияющем на передачу информации. Данная команда выполняется устройством аппаратно и не вызывает прерывания ПК.

По команде "Начать самоконтроль" осуществляется проверка аппаратуры терминала ОУ. Сюда включается сама устройство ТМК и другое оборудование, которое в данной системе может считаться частью терминала МК. Эта команда может быть использована как при инициализации системы, так и в процедурах восстановления после отказа МК. Устройство не содержит встроенных средств самоконтроля, поэтому указанная команда должна выполняться программно. После выполнения самоконтроля программа ОУ при обнаружении неисправности должна установить вызовом rtsetanswbits(RTFL) бит ОС "Неисправность терминала", а вызовом rtputcmddata задать слово встроенного контроля.

Команда "Подавить бит флага терминала" используется для установки соответствующего бита ОС в состояние "исправно", независимо от действительного состояния адресуемого ОУ. Это позволяет предотвратить повторяющиеся прерывания в системе обработки ошибок и восстановления, когда отказ обнаружен и система преобразована. Выдача этой команды блокирует передачу последующих сообщений об ошибке, которые должны были бы возникнуть при использовании признака неисправности терминала в каждом последующем ОС. Данная команда выполняется устройством аппаратно и не вызывает прерывания в ПК.

По команде "Отменить подавление бита флага терминала" снимается действие блокировки, разрешая биту признака неисправности терминала в ОС отображать действительное состояние терминала. Данная команда выполняется устройством аппаратно и не вызывает прерывания в ПК.

Команды "Блокировать передатчик" и "Разблокировать передатчик" используются в системах с двойным резервированием ЛПИ. Получив такую команду по одной ЛПИ, ОУ блокирует или снимает блокировку передатчика другой ЛПИ. Ответное слово передается в КК через линию, по которой поступила команда. Данные команды выполняются устройством аппаратно и не вызывают прерывания в ПК. При выполнении команд формируется внутриплатный сигнал блокировки передатчика резервного канала, который должен подаваться на вход блокировки соответствующей микросхемы передатчика.

По команде "Передать слово встроенного контроля" ОУ отвечает ОС с присоединенным словом данных, которое является кодом результата встроенного контроля. Эта команда позволяет КК получить информацию о возможной причине неисправности ОУ. Команда не должна использоваться для получения информации о неисправности подсистемы. Данная команда обычно используется КК, если в ОС появляется установленный бит "Неисправность терминала". Само слово встроенного контроля должно быть предварительно задано программой ОУ с помощью функции rtputcmddata.