УТВЕРЖДАЮ

Генеральный директор

ООО «НОВОМАР»

_____ Т.В. Буга

«____»_____2023 г.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«PCI ДРАЙВЕР MIL1553UD»

Модулей
“PCIe–1553UDx”
“XMC–1553UDx”
“CPCIS–1553UDx”
“mPCIe–1553UDx”

(ОС LINUX)
Руководство программиста

ЛИСТ УТВЕРЖДЕНИЯ

RU.МСКЮ.20102-01 33 01-ЛУ

От	Инженер-программист
_____	_____ _____
«____»_____2023 г.	«____»_____2023 г.

2023

Из	*Под*	*Дат*

Литера

Подп. и	
Инв. № дубл	
Взам. инв. №	
Подп. и	
Инв. № подл	

Утвержден

RU.МСКЮ.20102-01 33 01-ЛУ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«PCI ДРАЙВЕР MIL1553UD»

Модулей
“PCIe–1553UDx”
“XMC–1553UDx”
“CPCIS–1553UDx”
“mPCIe–1553UDx”

(ОС LINUX)

Руководство программиста

RU.МСКЮ.20102-01 33 01

Листов 32

2023

АННОТАЦИЯ

В книге описываются технологические принципы, использованные в программном обеспечении «PCI ДРАЙВЕР MIL1553UD» (ОС Linux и Astra Linux), для работы модулей PCIe-1553UDx", "XMC-1553UDx", "CPCIS-1553UDx" и "mPCIe-1553UDx" в сети МКИО ГОСТ Р 52070-2003. В частности, рассмотрены функциональное назначение и область применения, условия выполнения.

СОДЕРЖАНИЕ

1 НАЗНАЧЕНИЕ ПРОГРАММЫ

1.1 ДРАЙВЕР MIL1553UD

Программное обеспечение «PCI ДРАЙВЕР MIL1553UD» (далее – драйвер) обеспечивает возможность управления PCI-устройствами “PCIe-1553UDx”, “XMC-1553UDx”, “CPCIS-1553UDx” и “mPCIe-1553UDx” (далее MIL1553UD).

MIL1553UD (1-4-х канальный контроллер интерфейса МКИО – ГОСТ Р 52070-2003).

Драйвер обеспечивает выполнение следующих основных задач:

– определение и инициализация устройств на шине PCI;

– инициализация символьных устройств каналов (1-4) для обеспечения взаимодействия из юзерспейс пространства;

– реализация команд управления каналами в режимах КШ, ОУ, МШ, МША.

2 УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

2.1 ДРАЙВЕР MIL1553UD

Драйвер является модулем ядра и предназначен для функционирования в ОС Linux и Astra Linux. Перед использованием драйвера необходимо его собрать и установить.

2.1.1 Инструкция по сборке и настройке драйвера

Проект драйвера можно собирать с помощью командной строки и утилиты make.

Предварительно установите пакеты с заголовочными файлами для ядра и утилиты для сборки:

```
sudo apt-get install linux-headers-$(uname –r) build-essential libelf-dev
```

Затем с помощью командной строки и утилиты make:

- открыть терминал в папке с проектом, в терминале выполнить: `make`
- для установки драйвера в автозагрузку: `sudo make install`
- для запуска установленного драйвера: `sudo insmod mil1553ud_driver.ko`
- для останова работающего драйвера: `sudo rmmod mil1553ud_driver`

Также доступны команды

- для очистки проекта: `make clean`
- для удаления драйвера: `sudo make uninstall`
- для проверки работает ли драйвер в данный момент: `sudo lsmod | grep mil1553ud_driver`

3 ХАРАКТЕРИСТИКА ПРОГРАММЫ

3.1 ДРАЙВЕР MIL1553UD

Драйвер является модулем ядра ОС Linux, разработан на языке С, после сборки представляет собой исполняемый объектный модуль с именем «mil1553ud_driver.ko».

Взаимодействие с каналами МКИО осуществляется через символьные устройства поканально, которые расположение в каталоге «/dev».

Шаблон имени символьного устройства (канала): «mil1553dev-N-ch-M», где N — порядковый номер устройства, M — порядковый номер канала данного устройства.

Нумерация плат (параметр N) с 0, нумерация каналов (параметр M) с 0.

Взаимодействие с каналами происходит посредством ioctl-команд.

Рекомендуется использовать библиотеку взаимодействия для работы с драйвером!

3.1.1 Настройка в режим КШ

Настройка канала в режим КШ осуществляется с помощью следующий IOCTL-команд:

```
IOCTL_SWITCH_MODE    // установить режим КШ
IOCTL_SWITCH_BUS // установить шину (А или Б)
IOCTL_BC_RAM // составляем и записываем микропрограмму
IOCTL_ENABLE_DMA // включаем DMA
IOCTL_DEV_CHANNEL_WORK // разрешаем работу канала
IOCTL_WRITE_REG_BIT_MASK // устанавливаем бит BCSTRT
...
// разбор принятых данных
IOCTL_GET_NBLOCK_RAW_DMA
IOCTL_READ_BLOCKS_RAW_DMA
```

3.1.2 Настройка в режим ОУ

Настройка канала в режим ОУ осуществляется с помощью следующих IOCTL-команд:

IOCTL_SWITCH_MODE // установить режим ОУ
IOCTL_SET_ADDRESS // установить адрес ОУ (1-30)
IOCTL_SWITCH_BUS // установить шину (А или Б)
IOCTL_RT_BUF_MAN_EN // задать маску разрешённых подадресов на приём
IOCTL_RT_BUF_MAN_EN // задать маску разрешённых подадресов на передачу
IOCTL_SET_RT_TR_MODE // выбрать режим работы буферов
IOCTL_SET_RT_BUF_READY // задать готовность буферов подаресов к отправке
IOCTL_ENABLE_DMA // включаем DMA
IOCTL_DEV_CHANNEL_WORK // разрешаем работу канала
...
// разбор принятых данных
IOCTL_GET_NBLOCK_RAW_DMA
IOCTL_READ_BLOCKS_RAW_DMA

4 ОБРАЩЕНИЕ К ПРОГРАММЕ

4.1 ДРАЙВЕР MIL1553UD

Взаимодействие с каналами происходит посредством ioctl-команд, описание команд и типы данных представлены в файле «mil1553ud_cmd.h», реализация функций ioctl-команд представлена в файле «mil1553ud_ioctl.c».

Рекомендуется использовать библиотеку взаимодействия для работы с драйвером!

Пример исходного кода программы-примера приведён в документации к библиотеке взаимодействия.

Ниже описан внутренний интерфейс драйвера – ioctl команды, вызов которых осуществляется в реализации функций библиотеки.

4.1.1 Запись в регистр - IOCTL_WRITE_REG

Позволяет записать значение в заданный регистр управления.

Описание параметров команды приведено на рисунке 1.

```
#define IOCTL_WRITE_REG _IOW(IOC_MAGIC, 0,  SADDR_DATA)

/// \brief запись/чтение регистров
typedef struct {
    /// \brief адрес регистра (смещение в соотв. со спецификацией)
    unsigned long daddr;
    /// \brief значение регистра
    unsigned int data;
} SADDR_DATA;
```

Рисунок 1 – Листинг команды

4.1.2 Запись в регистр заданных бит - IOCTL_WRITE_REG_BIT_MASK

Позволяет изменить отдельные биты в заданном регистре управления.

Описание параметров команды приведено на рисунке 2.

```
#define IOCTL_WRITE_REG_BIT_MASK _IOW(IOC_MAGIC, 1,
SADDR_DATA_BIT_MASK)

/// \brief запись в регистр заданных бит
typedef struct {
    /// \brief адрес регистра (смещение в соотв. со спецификацией)
    unsigned long daddr;
    /// \brief значение регистра
    unsigned int data;
    /// \brief битовая маска
    /// 1 - бит записывается из data, 0 - бит остаётся неизменным
    unsigned int mask;
} SADDR_DATA_BIT_MASK;
```

Рисунок 2 – Листинг команды

4.1.3 Чтение из регистра - IOCTL_READ_REG

Позволяет прочитать значение из заданного регистра управления.

Описание параметров команды приведено на рисунке 3.

```
#define IOCTL_READ_REG _IOWR(IOC_MAGIC,2,  SADDR_DATA)

/// \brief запись/чтение регистров
typedef struct {
    /// \brief адрес регистра (смещение в соотв. со спецификацией)
    unsigned long daddr;
    /// \brief значение регистра
    unsigned int data;
} SADDR_DATA;
```

Рисунок 3 – Листинг команды

4.1.4 Чтение подробной информации о плате и драйвере - IOCTL_VERSION

Позволяет считать подробную информацию о pci-плате.

Описание параметров команды приведено на рисунке 4.

```
#define IOCTL_VERSION _IOR(IOC_MAGIC, 31, VERSION)

/// \brief информация о драйвере и устройстве
typedef struct {
    /// \brief идентификатор устройства
    unsigned int device_id;
    /// \brief вендор устройства
    unsigned int vendor_id;
    /// \brief тип устройства (кол-во каналов)
    unsigned int type;
    /// \brief ревизия устройства
    char revision;
    /// \brief имя символьного устройства
    char dev_name[30];
    /// \brief значение минора
    int minor;
    /// \brief номер прерывания
    int irq;
    /// \brief размер DMA буфера
    long size_dma;
    /// \brief виртуальный адрес DMA буфера
    void* addr_dma_virt;
    /// \brief адрес bar-пространства
    unsigned int pci_bars;
    /// \brief виртуальный адрес bar-пространства
    void* addr_bar_virt;
} VERSION;
```

Рисунок 4 – Листинг команды

4.1.5 Чтение версии драйвера - IOCTL_VERSION_DRIVER

Позволяет считать версию и дату драйвера.

Описание параметров команды приведено на рисунке 5.

```
#define IOCTL_VERSION_DRIVER _IOR(IOC_MAGIC, 40, unsigned int)

/// \remark информация о дате и версии
/// \brief старший номер версии
#define DRIVER_MAJOR_VER                2
/// \brief младший номер версии
#define DRIVER_MINOR_VER                0
/// \brief день создания
#define DRIVER_DATE_DAY                 16
/// \brief месяц создания
#define DRIVER_DATE_MONTH               06
/// \brief год создания
#define DRIVER_DATE_YEAR                19
/// \brief закодированная дата и версия
///  31..28 - major_ver; 27..24 - minor_ver; 23..16 - day; 15..8 -
month; 7..0 - year;
#define DRIVER_DATE_N_VERSION           ((DRIVER_MAJOR_VER & 0xF) <<
28) | ((DRIVER_MINOR_VER & 0xF) << 24) | ((DRIVER_DATE_DAY & 0xFF)
<< 16) | ((DRIVER_DATE_MONTH & 0xFF) << 8) | (DRIVER_DATE_YEAR &
0xFF)
```

Рисунок 5 – Листинг команды

4.1.6 Чтение pci-локации платы - IOCTL_PCI_LOCATION

Позволяет прочитать информацию о pci-локации платы. В случае подключения по другому интерфейсу, будет возвращён код ошибки.

Описание параметров команды приведено на рисунке 6.

```
#define IOCTL_PCI_LOCATION _IOR(IOC_MAGIC, 41, PCI_LOCATION)

/// \brief информация о локации на шине pci
typedef struct {
    /// \brief  номер шины
    unsigned int pBus;
    /// \brief  номер слота
    unsigned int pSlot;
} PCI_LOCATION;
```

Рисунок 6 – Листинг команды

4.1.7 Сброс указателя dma канала - IOCTL_CLEAR_DMA

Позволяет обнулить DMA_INDEX и программные указатели чтения/записи.

Не имеет параметров.

4.1.8 Включить dma устройства - IOCTL_ENABLE_DMA

Разрешает работу DMA.

Не имеет параметров.

4.1.9 Выключить dma устройства - IOCTL_DISABLE_DMA

Запрещает работу DMA.

Не имеет параметров.

4.1.10 Получить количество 128 байтных блоков из DMA - IOCTL_GET_NBLOCK_RAW_DMA

Позволяет получить количество готовых для чтения блоков из буфера DMA.

Описание параметров команды приведено на рисунке 7.

```
#define IOCTL_GET_NBLOCK_RAW_DMA _IOWR(IOC_MAGIC,7,  unsigned int)
```

Рисунок 7 – Листинг команды

4.1.11 Считать заданное количество 128 байтных блоков из DMA - IOCTL_READ_BLOCKS_RAW_DMA

Позволяет получить данные (блоки) из буфера DMA.

Описание параметров команды приведено на рисунке 8-а.

```
#define IOCTL_READ_BLOCKS_RAW_DMA _IOWR(IOC_MAGIC,6,
DMA_READ_BLOCK)

/// \brief считываемый блок DMA, кратный 128 байтам
typedef struct {
    /// \brief количество 128 байтных блоков
    unsigned int countBlocks;
    /// \brief длина data в байтах
    unsigned int length;
    /// \brief данные блоков в сыром виде
    unsigned char* data;
} DMA_READ_BLOCK;
```

Рисунок 8-а – Листинг команды

4.1.12 Считать заданное количество 128 байтных блоков из DMA - IOCTL_READ_BLOCKS_RAW_DMA_FIX_COUNT

Позволяет за один вызов считать данные (блоки) из буфера DMA устройства в предвыделенный пользовательский буфер, а также узнать количество оставшихся блоков в буфере после выполненного чтения.

Описание параметров команды приведено на рисунке 8-б.

```
#define IOCTL_READ_BLOCKS_RAW_DMA_FIX_COUNT _IOWR(IOC_MAGIC,60,
MIL1553_DMA_FIX_BLOCK

/// \brief контейнер блоков ДМА (на 256 блоков - максимум)
typedef struct {
    unsigned int currentBlocks;      ///< фактическое количество
блоков ДМА (не более mallocedBlocks)
    unsigned int nextBlocks;         ///< кол-во блоков, готовых к
чтению в целевом устройстве (кольцевом буфере ДМА девайса)
    unsigned int mallocedBlocks;     ///< выделенное место в data
    unsigned char* data;    ///< блоки ДМА в сыром виде (кратные 128
байтам)
} MIL1553_DMA_FIX_BLOCK;
```

Рисунок 8-б – Листинг команды

4.1.13 Управление прерываниями INTERRUPT_MASK - IOCTL_MANAGE_INTERRUPT

Позволяет управлять прерываниями.

Описание параметров команды приведено на рисунке 9.

```
#define IOCTL_MANAGE_INTERRUPT _IOW(IOC_MAGIC, 15, INTERRUPT_MAN)

/// \brief управление маскированием перываний поканально
typedef struct {
    /// \brief состояние прерывания от контроллера flash
    unsigned char int_flash;
    /// \brief состояние прерывания от КШ
    unsigned char int_bc;
    /// \brief состояние прерывания от ОУ при приёме данных
    unsigned char int_rt_ren;
    /// \brief состояние прерывания от ОУ при отправке данных
    unsigned char int_rt_ten;
    /// \brief состояние прерывания при заполнении 1/18
    unsigned char int_qdat;
    /// \brief состояние прерывания при заполнении 1/2
    unsigned char int_hdat;
    /// \brief состояние прерывания счётчика данных контроллера MIL
    unsigned char int_data_cnt_en;
    /// \brief состояние прерывания интервального таймера MIL
    unsigned char int_timeout_itven;
    /// \brief состояние прерывания абсолютного таймера MIL
    unsigned char int_timeout_absen;
} INTERRUPT_MAN;

/// \remark константы для управления прерываниями
/// 0 - не меняется значение, 1 - включить прерывание, 2 - выключить
прерывание
/// \brief не изменять бит прерывания
#define INTERRUPT_MAN_NO_CHANGE         0
/// \brief включить бит прерывания
#define INTERRUPT_MAN_ON                1
/// \brief выключить бит прерывания
#define INTERRUPT_MAN_OFF               2
```

Рисунок 9 – Листинг команды

4.1.14 Управление прерываниями по передаче из подадреса - IOCTL_MAN_IRQ_SUBADDRESS_TR

Позволяет управлять прерываниями по передаче из подадреса.

Описание параметров команды приведено на рисунке 10.

```
#define IOCTL_MAN_IRQ_SUBADDRESS_TR _IOW(IOC_MAGIC, 38,
INTERRUPT_SUBADDRESS)

/// \brief управление маскированием прерываний по приёму/передачи
данных по подадресам
typedef struct {
    /// \brief состояние прерывания (вкл - 1 / выкл - 2)
    unsigned char rt_int;
    /// \brief маскирование прерываний по подадресам 1й бит - 1й
подадрес и т.д.
    unsigned int subaddress_mask;
} INTERRUPT_SUBADDRESS;

/// \brief включить бит прерывания
#define INTERRUPT_MAN_ON                1
/// \brief выключить бит прерывания
#define INTERRUPT_MAN_OFF               2
```

Рисунок 10 – Листинг команды

4.1.15 Управление прерываниями по приёму в подадрес - IOCTL_MAN_IRQ_SUBADDRESS_RCV

Позволяет управлять прерываниями по приёму в подадрес.

Описание параметров команды приведено на рисунке 11.

```
define IOCTL_MAN_IRQ_SUBADDRESS_RCV _IOW(IOC_MAGIC, 39,
INTERRUPT_SUBADDRESS)

/// \brief управление маскированием прерываний по приёму/передачи
данных по подадресам
typedef struct {
    /// \brief состояние прерывания (вкл - 1 / выкл - 2)
    unsigned char rt_int;
    /// \brief маскирование прерываний по подадресам 1й бит - 1й
подадрес и т.д.
    unsigned int subaddress_mask;
} INTERRUPT_SUBADDRESS;

/// \brief включить бит прерывания
#define INTERRUPT_MAN_ON                1
/// \brief выключить бит прерывания
#define INTERRUPT_MAN_OFF               2
```

Рисунок 11 – Листинг команды

4.1.16 Выбор режима работы канала - IOCTL_SWITCH_MODE

Позволяет выбрать режим работы канала.

Описание параметров команды приведено на рисунке 12.

```
#define IOCTL_SWITCH_MODE _IOW(IOC_MAGIC, 4,  unsigned int)

/// \remark константы режимов работы устройства
/// \brief адресуемый монитор шины
#define MIL_MODE_MONITOR_ADRR           0x0
/// \brief контроллер шины
#define MIL_MODE_BUS_CONTR              0x1
/// \brief оконечное устройство
#define MIL_MODE_TERMINAL_DEV           0x2
/// \brief НЕадресуемый монитор шины
#define MIL_MODE_MONITOR                0x3
```

Рисунок 12 – Листинг команды

4.1.17 Выбор шины канала - IOCTL_SWITCH_BUS

Позволяет выбрать шину канала.

Описание параметров команды приведено на рисунке 13.

```
#define IOCTL_SWITCH_BUS _IOW(IOC_MAGIC, 33, unsigned int)

/// \remark константы выбора шины канала
/// \brief включить шину A
#define MIL_BUS_A_EN                    0x1
/// \brief включить шину B
#define MIL_BUS_B_EN                    0x2
```

Рисунок 13 – Листинг команды

4.1.18 Управление работой канала - IOCTL_DEV_CHANNEL_WORK

Позволяет управлять работой канала.

Описание параметров команды приведено на рисунке 14.

```
#define IOCTL_DEV_CHANNEL_WORK _IOW(IOC_MAGIC, 34, unsigned int)

/// \remark константы вкл/выкл канала
/// \brief включить канал в работу
#define MIL_DEV_CHANNEL_ON              0x1
/// \brief выключить канал
#define MIL_DEV_CHANNEL_OFF             0x0
```

Рисунок 14 – Листинг команды

4.1.19 Установить режим ОУ - IOCTL_SET_RT_TR_MODE

Позволяет установить режим ОУ.

Описание параметров команды приведено на рисунке 15.

```
#define IOCTL_SET_RT_TR_MODE _IOW(IOC_MAGIC, 35, RT_TR_MODE)

/// \brief управление режимом работы ОУ по подадресам
typedef struct {
    /// \brief подадрес 1 - 30
    unsigned char subaddress;
    /// \brief режим
    unsigned int mode;
} RT_TR_MODE;

/// \remark константы для режима ОУ
/// \brief режим - программный
#define MIL_RT_MODE_PROG                0x0
/// \brief режим - аппаратный
```

Рисунок 15 – Листинг команды

4.1.20 Установить готовность буфера ОУ - IOCTL_SET_RT_TR_BUF_READY

Позволяет установить готовность буфера ОУ.

Описание параметров команды приведено на рисунке 16.

```
#define IOCTL_SET_RT_TR_BUF_READY _IOW(IOC_MAGIC, 37,
RT_TR_MODE_BUF)

/// \brief управление буферами 0 и 1
typedef struct {
    /// \brief подадрес 1 - 30
    unsigned char subaddress;
    /// \brief команда упр. буфером
    unsigned int cmd_buf;
} RT_TR_MODE_BUF;

/// \remark команды управления буферами cmd_buf
/// \brief буфер данных RTF_BUF0 и RTF_BUF1 - выкл
#define MIL_RT_BUF_0_OFF_1_OFF          0x0
/// \brief буфер данных RTF_BUF0 - выкл и RTF_BUF1 - вкл
#define MIL_RT_BUF_0_OFF_1_ON           0x1
/// \brief буфер данных RTF_BUF0 - вкл и RTF_BUF1 - выкл
#define MIL_RT_BUF_0_ON_1_OFF           0x2
/// \brief буфер данных RTF_BUF0 - вкл и RTF_BUF1 - вкл
#define MIL_RT_BUF_0_ON_1_ON            0x3
```

Рисунок 16 – Листинг команды

4.1.21 Управление разрешением буферов - IOCTL_RT_BUF_MAN_EN

Позволяет осуществлять управление разрешением буферов.

Описание параметров команды приведено на рисунке 17.

```
#define IOCTL_RT_BUF_MAN_EN _IOW(IOC_MAGIC, 36, RT_BUF_MAN_EN)

/// \brief управление подадресами в режиме ОУ
typedef struct {
    /// \brief направление (приём/передача)
    unsigned int direction;
    /// \brief маскирование (выбор) по подадресам 1й бит - 1й
подадрес и т.д.
    /// 1 - бит изменяем, 0 - бит не изменяем
    unsigned int subaddress_mask;
    /// \brief маскирование (выбор) действий по подадресам 1й бит -
1й подадрес и т.д.
    /// 1 - вкл, 0 - выкл
    unsigned int action_mask;
} RT_BUF_MAN_EN;

/// \remark направление передачи direction
/// \brief направление - передача
#define MIL_RT_BUF_TRANSMIT             0x0
/// \brief направление - передача
#define MIL_RT_BUF_RECEIVE              0x1
```

Рисунок 17 – Листинг команды

4.1.22 Установить адрес ОУ - IOCTL_SET_ADDRESS

Позволяет установить адрес ОУ.

Описание параметров команды приведено на рисунке 18.

```
#define IOCTL_SET_ADDRESS _IOW(IOC_MAGIC, 19, unsigned char)
```

Рисунок 18 – Листинг команды

4.1.23 Установить таймауты ОУ - IOCTL_SET_TIMER_RTBM_CONF_REG_PCI

Позволяет установить таймауты ОУ.

Описание параметров команды приведено на рисунке 19.

```
#define IOCTL_SET_TIMER_RTBM_CONF_REG_PCI _IOW(IOC_MAGIC, 20,
TIMER_TIMEOUT)

/// \brief управление таймаутами таймеров
typedef struct {
    /// \brief значение RT_TCK
    unsigned int tck;
    /// \brief значение RT_TRCK
    unsigned int trck;
    /// \brief значение RT_RCVCK
    unsigned int rcvck;
} TIMER_TIMEOUT;
/// \remark константы для timer timeout RCVCK
#define MIL_T_RCVCK_17MKS               0x0
#define MIL_T_RCVCK_60MKS               0x1
#define MIL_T_RCVCK_85MKS               0x2
#define MIL_T_RCVCK_110MKS              0x3
/// \remark константы для timer timeout TRCK
#define MIL_T_TRCK_6MKS                 0x0
#define MIL_T_TRCK_8MKS                 0x1
#define MIL_T_TRCK_11MKS                0x2
#define MIL_T_TRCK_13MKS                0x3
#define MIL_T_TRCK_18MKS                0x4
#define MIL_T_TRCK_61MKS                0x5
#define MIL_T_TRCK_86MKS                0x6
#define MIL_T_TRCK_111MKS               0x7
/// \remark константы для timer timeout TCK
#define MIL_T_TCK_OFF                   0x0
#define MIL_T_TCK_1MKS                  0x1
#define MIL_T_TCK_2MKS                  0x2
#define MIL_T_TCK_4MKS                  0x3
#define MIL_T_TCK_8MKS                  0x4
#define MIL_T_TCK_16MKS                 0x5
#define MIL_T_TCK_32MKS                 0x6
#define MIL_T_TCK_64MKS                 0x7
```

Рисунок 19 – Листинг команды

4.1.24 Установить таймауты КШ - IOCTL_SET_TIMER_BC_CONF_REG_PCI

Позволяет установить таймауты КШ.

Описание параметров команды приведено на рисунке 20.

```
#define IOCTL_SET_TIMER_BC_CONF_REG_PCI _IOW(IOC_MAGIC, 20,
TIMER_TIMEOUT)

/// \brief управление таймаутами таймеров
typedef struct {
    /// \brief значение RT_TCK
    unsigned int tck;
    /// \brief значение RT_TRCK
    unsigned int trck;
    /// \brief значение RT_RCVCK
    unsigned int rcvck;
} TIMER_TIMEOUT;
/// \remark константы для timer timeout RCVCK
#define MIL_T_RCVCK_17MKS               0x0
#define MIL_T_RCVCK_60MKS               0x1
#define MIL_T_RCVCK_85MKS               0x2
#define MIL_T_RCVCK_110MKS              0x3
/// \remark константы для timer timeout TRCK
#define MIL_T_TRCK_6MKS                 0x0
#define MIL_T_TRCK_8MKS                 0x1
#define MIL_T_TRCK_11MKS                0x2
#define MIL_T_TRCK_13MKS                0x3
#define MIL_T_TRCK_18MKS                0x4
#define MIL_T_TRCK_61MKS                0x5
#define MIL_T_TRCK_86MKS                0x6
#define MIL_T_TRCK_111MKS               0x7
/// \remark константы для timer timeout TCK
#define MIL_T_TCK_OFF                   0x0
#define MIL_T_TCK_1MKS                  0x1
#define MIL_T_TCK_2MKS                  0x2
#define MIL_T_TCK_4MKS                  0x3
#define MIL_T_TCK_8MKS                  0x4
#define MIL_T_TCK_16MKS                 0x5
#define MIL_T_TCK_32MKS                 0x6
#define MIL_T_TCK_64MKS                 0x7
```

Рисунок 20 – Листинг команды

4.1.25 Записать подадрес на отправку - IOCTL_WR_BLOCK_BUF_SUBADR

Позволяет записать данные в подадрес на отправку.

Описание параметров команды приведено на рисунке 21.

```
#define IOCTL_WR_BLOCK_BUF_SUBADR _IOW(IOC_MAGIC, 27,
RT_TR_BUF_SUBADDR)

/// \brief буфер ОУ на передачу подадреса
typedef struct {
    /// \brief номер буфера
    unsigned char num_buf;
    /// \brief подадрес (1-30)
    unsigned char subaddress;
    /// \brief подадрес (32 слова данных)
    unsigned short data_words[32];
} RT_TR_BUF_SUBADDR;

/// \remark номер буфера num_buf
/// \brief отправной буфер данных RTF_BUF0
#define MIL_RT_BUF0                    0
/// \brief отправной буфер данных RTF_BUF1
#define MIL_RT_BUF1                    1
```

Рисунок 21 – Листинг команды

4.1.26 Прочитать подадрес на отправку - IOCTL_RD_BLOCK_BUF_SUBADR

Позволяет прочитать данные из подадреса на отправку.

Описание параметров команды приведено на рисунке 22.

```
#define IOCTL_RD_BLOCK_BUF_SUBADR _IOW(IOC_MAGIC, 28,
RT_TR_BUF_SUBADDR)

/// \brief буфер ОУ на передачу подадреса
typedef struct {
    /// \brief номер буфера
    unsigned char num_buf;
    /// \brief подадрес (1-30)
    unsigned char subaddress;
    /// \brief подадрес (32 слова данных)
    unsigned short data_words[32];
} RT_TR_BUF_SUBADDR;

/// \remark номер буфера num_buf
/// \brief отправной буфер данных RTF_BUF0
#define MIL_RT_BUF0                    0
/// \brief отправной буфер данных RTF_BUF1
#define MIL_RT_BUF1                    1
```

Рисунок 22 – Листинг команды

4.1.27 Запись блока в BC RAM - IOCTL_WRITE_BC_RAM

Позволяет записать блок данных для записи в BC_RAM (область микропрограммы для КШ).

Описание параметров команды приведено на рисунке 23.

```
#define IOCTL_WRITE_BC_RAM _IOW(IOC_MAGIC, 29, BC_RAM_BLOCK)

/// \brief блок данных для записи в BC_RAM (область микропрограммы
для КШ)
typedef struct {
    /// \brief тип инструкций
    /// INSTR, OPERATION, DATA
    unsigned int type;
    /// \brief смещение от начала буфера в 32-х разрядных словах
    /// INSTR - max 4095, OPERATION - max 4095, DATA - max 8191
    unsigned int shift;
    /// \brief размер поля dwords в 32-х разрядных словах
    unsigned int length;
    /// \brief данные (массив 32-х разрядных слов)
    unsigned int* dwords;
} BC_RAM_BLOCK;

/// \remark константы для команды записи/чтения BC RAM

#define MIL_BC_RAM_TYPE_INSTRUCTION     0x0
#define MIL_BC_RAM_TYPE_OPERATION       0x1
#define MIL_BC_RAM_TYPE_DATA            0x2
```

Рисунок 23 – Листинг команды

4.1.28 Чтение блока из BC RAM - IOCTL_WRITE_BC_RAM

Позволяет прочитать блок данных из BC_RAM (область микропрограммы для КШ).

Описание параметров команды приведено на рисунке 24.

```
#define IOCTL_READ_BC_RAM _IOW(IOC_MAGIC, 30, BC_RAM_BLOCK)

/// \brief блок данных для записи в BC_RAM (область микропрограммы
для КШ)
typedef struct {
    /// \brief тип инструкций
    /// INSTR, OPERATION, DATA
    unsigned int type;
    /// \brief смещение от начала буфера в 32-х разрядных словах
    /// INSTR - max 4095, OPERATION - max 4095, DATA - max 8191
    unsigned int shift;
    /// \brief размер поля dwords в 32-х разрядных словах
    unsigned int length;
    /// \brief данные (массив 32-х разрядных слов)
    unsigned int* dwords;
} BC_RAM_BLOCK;

/// \remark константы для команды записи/чтения BC RAM

#define MIL_BC_RAM_TYPE_INSTRUCTION     0x0
#define MIL_BC_RAM_TYPE_OPERATION       0x1
#define MIL_BC_RAM_TYPE_DATA            0x2
```

Рисунок 24 – Листинг команды

4.1.29 Получить кол-во блоков прерываний - IOCTL_WR_BLOCK_BUF_SUBADR

Позволяет получить кол-во блоков с информацией о возникших прерываниях из кольцевого буфера.

Описание параметров команды приведено на рисунке 25.

```
#define IOCTL_GET_COUNT_INTERRUPT_BLOCKS _IOWR(IOC_MAGIC,50,
unsigned int)
```

Рисунок 25 – Листинг команды

4.1.30 Считать блоки прерываний - IOCTL_WR_BLOCK_BUF_SUBADR

Позволяет считать заданное кол-во блоков из кольцевого буфера.

Описание параметров команды приведено на рисунке 26.

```
#define IOCTL_READ_INTERRUPT_BLOCKS _IOWR(IOC_MAGIC,51,
INTERRUPT_BLOCK_BUFFER)

/// \brief блок данных для чтения накопленных прерываний
typedef struct {
    /// \brief кол-во запришиваемых блоков
    unsigned int        count_blocks;
    /// \brief собственно указатель на массив блоков
    struct block_info*  blocks;
} INTERRUPT_BLOCK_BUFFER;

/// \brief блок информации о прерывании
struct block_info {
    /// \brief значение таймера в момент прерывани
    unsigned int    free_timer_value;
    /// \brief маска прерываний (собственно по ней поймём какие в
этот момент времени были прерывания)
    /// 0 - INT_HDAT, 1 - INT_QDAT, 2 - RT_INT_SADDR, 3 -
RT_INT_MC_ERR,
    /// 4 - INT_BC, 5 - INT_FLASH, 6 - INT_DATA_CNT_EN,
    /// 7 - INT_TIMEOUT_ITVEN, 8 - INT_TIMEOUT_ABSEN
    unsigned int    interrupt_ch;
    /// \brief регистр HW_STAT_REG1
    unsigned int    hw_stat_reg1;
    /// \brief регистр HW_STAT_REG2
    unsigned int    hw_stat_reg2;
};

#define BLINF_INT_HDAT          0
#define BLINF_INT_QDAT          1
#define BLINF_RT_INT_SADDR      2
#define BLINF_RT_INT_MC_ERR     3
#define BLINF_INT_BC            4
#define BLINF_INT_FLASH         5
#define BLINT_INT_DATA_CNT_EN   6
#define BLINT_TIMEOUT_ITVEN     7
#define BLINT_TIMEOUT_ABSEN     8
```

Рисунок 26 – Листинг команды

4.1.31 Считать входной подадрес - IOCTL_GET_RCV_SUBADR_DATA

Позволяет считать данные из входного подареса.

Описание параметров команды приведено на рисунке 27.

```
#define IOCTL_GET_RCV_SUBADR_DATA _IOWR(IOC_MAGIC,52,
SUBA_DATA_BLOCK)

/// \brief информация о подадресе
typedef struct {
    /// \brief подадрес (1-30)
    unsigned char sa;
    /// \brief слова данных
    unsigned short words[32];
} SUBA_DATA_BLOCK;
```

Рисунок 27 – Листинг команды

4.1.32 Получить номер канала - IOCTL_GET_NUMBER_CH

Позволяет получить номер канала.

Описание параметров команды приведено на рисунке 28.

```
#define IOCTL_GET_NUMBER_CH _IOWR(IOC_MAGIC,55,  unsigned int)

/// \remark номера каналов
/// \brief канал 1
#define CH_NUM_1            0
/// \brief канал 2
#define CH_NUM_2            1
/// \brief канал 3
#define CH_NUM_3            2
/// \brief канал 4
#define CH_NUM_4            3
```

Рисунок 28 – Листинг команды

5 СООБЩЕНИЯ

5.1 ДРАЙВЕР MIL1553UD

В процессе работы драйвер сохраняет отладочную информацию, которую можно увидеть с помощью команды dmesg, набранную в терминале ОС Linux.

5.1.1 Обработка ошибок IOCTL-команд

Описание кодов ошибок приведено на рисунке 83.

```
/// \brief нет ошибок
#define GOOD                            0
/// \brief ошибка
#define ERROR                           -1000
/// \brief ошибка адреса регистра
#define ERR_BAD_ADDRESS                 -1001
/// \brief ошибка номера канала
/// внутренняя ошибка в драйвере
#define ERR_BAD_CHANNEL                 -1002
/// \brief ошибка копирования данных в юзерспейс
#define ERR_COPY_TO_USER                -1003
/// \brief ошибка аргумента - mode
#define ERR_BAD_ARG_MODE                -1004
/// \brief ошибка аргумента - subaddress
#define ERR_BAD_ARG_SA                  -1005
/// \brief ошибка аргумента - direction
#define ERR_BAD_ARG_DIRECTION           -1006
/// \brief ошибка аргумента - tck
#define ERR_BAD_ARG_TCK                 -1007
/// \brief ошибка аргумента - trck
#define ERR_BAD_ARG_TRCK                -1008
/// \brief ошибка аргумента - rcvck
#define ERR_BAD_ARG_RCVCK               -1009
/// \brief ошибка аргумента - numbuf
#define ERR_BAD_ARG_NUMBUF              -1010
/// \brief ошибка аргумента - typr
#define ERR_BAD_ARG_TYPE                -1011
/// \brief ошибка выделения памяти
/// внутренняя ошибка драйвера
#define ERR_BAD_ALLOC_MEM               -1012
```

Рисунок 83 – Листинг

СПИСОК СОКРАЩЕНИЙ

ПО – программное обеспечение;

МКИО – интерфейс по ГОСТ Р 52070-2003;

КШ – контроллер шины;

ОУ – оконечное устройство;

МШ – монитор шины;

МША – монитор шины адресный;

DMA – direct memory access (прямой доступ к памяти);

Лист регистрации

Номера листов					*Всего листов (страниц) в докум.*	*№ документа*	*Входящий № сопроводительного докум. и дата*	*Подп.*	*Дата*
Изм.	*измененных*	*замененных*	*новых*	*аннулированных*					