УТВЕРЖДАЮ

Генеральный директор

ООО «НОВОМАР»

_____ Т.В. Буга

«____»_____2023 г.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«ДРАЙВЕР MIL1553UD»

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

(ОС WINDOWS)

Руководство программиста

ЛИСТ УТВЕРЖДЕНИЯ

RU.МСКЮ.20201-04 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.20201-04 33 01-ЛУ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«ДРАЙВЕР MIL1553UD»

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

(ОС WINDOWS)

Руководство программиста

RU.МСКЮ.20201-04 33 01

Листов 42

2023

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

Из	Под	Дат

Литера

АННОТАЦИЯ

В книге описываются технологические принципы, использованные в программном обеспечении «ДРАЙВЕР MIL1553UD» (ОС WINDOWS), для работы модулей PCIe-1553UDx", "XMC-1553UDx", "CPCIS-1553UDx", "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 «ДРАЙВЕР MIL1553UD» (далее – драйвер) обеспечивает возможность управления PCI-устройствами “PCIe-1553UDx”, “XMC-1553UDx”, “CPCIS-1553UDx”, “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 устройства 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.

1.2 ЗАГОЛОВОЧНЫЕ ФАЙЛЫ MIL1553UD

Описывают используемые драйвером коды запросов IOCTL и структуры данных.

2 УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

2.1 УСТАНОВКА ДРАЙВЕРА MIL1553UD

Драйвер является модулем ядра и предназначен для функционирования в ОС семейства Windows - 7, 8, 8.1, 10, поддерживаются архитектуры x86 и amd64.

Установка и удаление драйвера производятся через диспетчер устройств ОС Windows.

3 ХАРАКТЕРИСТИКА ПРОГРАММЫ

3.1 ДРАЙВЕР MIL1553UD

Драйвер является модулем ядра ОС Windows, разработан на языке С.

Находящиеся под управлением драйвера устройства доступны в системе по абсолютному пути \Device\mil1553ud_n, где n - порядковый номер устройства начиная с 0.

Драйвер реализует интерфейс класса GUID_DEVINTERFACE_MIL1553 (3bd2b180-d211-4d88-8f46-b73845cf38fb), что позволяет обнаружить все устройства в системе с помощью функции SetupDiGetClassDevs (см. документацию setupapi).

Взаимодействие с драйвером происходит посредством запросов ioctl.

3.1.1 Настройка в режим КШ

Настройка канала в режим КШ осуществляется с помощью следующией последовательности 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 ДМА
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 следующей последовательности 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 ДМА

IOCTL_DEV_CHANNEL_WORK // разрешаем работу канала

...

// разбор принятых данных

IOCTL_GET_NBLOCK_RAW_DMA

IOCTL_READ_BLOCKS_RAW_DMA

3.2 ЗАГОЛОВОЧНЫЕ ФАЙЛЫ MIL1553UD

Заголовочные файлы описывают используемые драйвером коды запросов IOCTL и структуры данных и предназначены для разработки на языках C и C++:

- regs.h
- info.h;
- ioctl_def.h;

Для использования заголовочных файлов в проекте необходимо включить вышеназванные файлы в состав разрабатываемого проекта.

4 ОБРАЩЕНИЕ К ПРОГРАММЕ

4.1 ДРАЙВЕР MIL1553UD

Взаимодействие с каналами происходит посредством ioctl-команд, описание команд и типы данных представлены в файле «ioctl_def.h».

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  CTL_CODE(FILE_DEVICE_CONTROLLER, 0,
METHOD_BUFFERED, FILE_ANY_ACCESS)

//входные данные
typedef struct {
    // адрес регистра (смещение в соотв. со спецификацией)
    UINT32 daddr;
    // значение регистра
    UINT32 data;
     //номер канала устройства
     UINT8 channel;
} 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   CTL_CODE(FILE_DEVICE_CONTROLLER,
1, METHOD_BUFFERED, FILE_ANY_ACCESS)

//входные данные

typedef struct {
     // адрес регистра (смещение в соотв. со спецификацией)
     UINT32 daddr;
     // значение регистра
     UINT32 data;
     // brief битовая маска
     // 1 - бит записывается из data, 0 - бит остаётся неизменным
     UINT32 mask;
     //номер канала устройства
     UINT8 channel;
} 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   CTL_CODE(FILE_DEVICE_CONTROLLER, 2,
METHOD_BUFFERED, FILE_ANY_ACCESS)

//входные данные
typedef struct {
    // адрес регистра (смещение в соотв. со спецификацией)
    UINT32 daddr;
    // неприменимо
    UINT32 data;
     //номер канала устройства
     UINT8 channel;
} SADDR_DATA;

//выходные данные
typedef struct {
    // адрес регистра (смещение в соотв. со спецификацией)
    UINT32 daddr;
    // значение регистра
    UINT32 data;
     //номер канала устройства
     UINT8 channel;
} 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
     CTL_CODE(FILE_DEVICE_CONTROLLER, 31, METHOD_BUFFERED,
FILE_ANY_ACCESS)

//выходные данные
typedef struct {
    // идентификатор устройства
    UINT32 device_id;
    // вендор устройства
    UINT32 vendor_id;
    // тип устройства (кол-во каналов)
    UINT32 type;
    // ревизия устройства
    char revision;
    // неприменимо
    char dev_name[MAXIMUM_FILENAME_LENGTH];
    // неприменимо
    int minor;
    // номер прерывания
    int irq;
    // размер ДМА буфера
    UINT32 size_dma;
    // неприменимо
    UINT32 pci_bars;
} 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
     CTL_CODE(FILE_DEVICE_CONTROLLER, 40, METHOD_BUFFERED,
FILE_ANY_ACCESS)

//выходные данные
/// \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 Сброс указателя dma канала - IOCTL_CLEAR_DMA

Позволяет обнулить DMA_INDEX и программные указатели чтения/записи.

Описание параметров команды приведено на рисунке 6.

```
#define IOCTL_CLEAR_DMA  CTL_CODE(FILE_DEVICE_CONTROLLER, 5,
METHOD_BUFFERED, FILE_ANY_ACCESS)  //сброс буфера

//входные данные
typedef struct _MODE_DATA{
     UINT8 channel; //номер канала
     UINT8 mode;    //неприменимо
} MODE_DATA, *PMODE_DATA;
```

Рисунок 6 – Листинг команды

4.1.7 Включить dma устройства - IOCTL_ENABLE_DMA

Разрешает работу DMA.

Описание параметров команды приведено на рисунке 7.

```
#define IOCTL_ENABLE_DMA CTL_CODE(FILE_DEVICE_CONTROLLER, 8,
METHOD_BUFFERED, FILE_ANY_ACCESS)  //вкл DMA

//входные данные
typedef struct _MODE_DATA{
     UINT8 channel; //номер канала
     UINT8 mode;    //неприменимо
} MODE_DATA, *PMODE_DATA;
```

Рисунок 7 – Листинг команды

4.1.8 Выключить dma устройства - IOCTL_DISABLE_DMA

Запрещает работу DMA.

Описание параметров команды приведено на рисунке 8.

```
#define IOCTL_DISABLE_DMA     CTL_CODE(FILE_DEVICE_CONTROLLER, 9,
METHOD_BUFFERED, FILE_ANY_ACCESS)  //выкл DMA

//входные данные
typedef struct _MODE_DATA{
     UINT8 channel; //номер канала
     UINT8 mode;    //неприменимо
} MODE_DATA, *PMODE_DATA;
```

Рисунок 8 – Листинг команды

4.1.9 Получить количество 128 байтных блоков из ДМА - IOCTL_GET_NBLOCK_RAW_DMA

Позволяет получить количество готовых для чтения блоков из буфера ДМА.

Описание параметров команды приведено на рисунке 9.

```
#define IOCTL_GET_NBLOCKS_RAW_DMA   CTL_CODE(FILE_DEVICE_CONTROLLER,
7, METHOD_BUFFERED, FILE_ANY_ACCESS)

//входные данные
typedef struct _DMA_COUNT_REQUEST{
     UINT8 channel; //канал
     UINT32 count;  //неприменимо
} DMA_COUNT_REQUEST, *PDMA_COUNT_REQUEST;

//выходные данные
typedef struct _DMA_COUNT_REQUEST{
     UINT8 channel; //канал
     UINT32 count;  //количество блоков
} DMA_COUNT_REQUEST, *PDMA_COUNT_REQUEST;
```

Рисунок 9 – Листинг команды

4.1.10 Считать заданное количество 128 байтных из ДМА - IOCTL_READ_BLOCKS_RAW_DMA

Позволяет получить данные (блоки) из буфера ДМА.

Описание параметров команды приведено на рисунке 10.

```
#define IOCTL_READ_BLOCKS_RAW_DMA
     CTL_CODE(FILE_DEVICE_CONTROLLER, 6, METHOD_BUFFERED,
FILE_ANY_ACCESS)

#define DMA_RAW_BLOCK_SIZE          128
#define DMA_SUPERBLOCK_SIZE         8192 * DMA_RAW_BLOCK_SIZE

//входные данные
/// \brief считываемый блок дма, кратный 128 байтам
typedef struct {
    /// \brief количество 128 байтных блоков
    UINT32 countBlocks;
    /// неприменимо
    UINT8 data[DMA_SUPERBLOCK_SIZE];
    //канал
    UINT8 channel;
} DMA_READ_BLOCK;

//выходные данные
/// \brief считываемый блок дма, кратный 128 байтам
typedef struct {
    /// \brief количество 128 байтных блоков
    UINT32 countBlocks;
    /// \brief данные блоков в сыром виде
    UINT8 data[DMA_SUPERBLOCK_SIZE];
    //канал
    UINT8 channel;
} DMA_READ_BLOCK;
```

Рисунок 10 – Листинг команды

4.1.11 Управление прерываниями INTERRUPT_MASK - IOCTL_MANAGE_INTERRUPT

Позволяет управлять прерываниями.

Описание параметров команды приведено на рисунке 11.

```
#define IOCTL_MANAGE_INTERRUPT     CTL_CODE(FILE_DEVICE_CONTROLLER,
16, METHOD_BUFFERED, FILE_ANY_ACCESS)

/// \brief управление маскированием перываний поканально
//входные данные
typedef struct {
    /// \brief состояние прерывания от контроллера flash
    UINT8 int_flash;
    /// \brief состояние прерывания от КШ
    UINT8 int_bc;
    /// \brief состояние прерывания от ОУ при приёме данных
    UINT8 int_rt_ren;
    /// \brief состояние прерывания от ОУ при отправке данных
    UINT8 int_rt_ten;
    /// \brief состояние прерывания при заполнении 1/18
    UINT8 int_qdat;
    /// \brief состояние прерывания при заполнении 1/2
    UINT8 int_hdat;
    /// \brief состояние прерывания счётчика данных контроллера MIL
    UINT8 int_data_cnt_en;
    /// \brief состояние прерывания интервального таймера MIL
    UINT8 int_timeout_itven;
    /// \brief состояние прерывания абсолютного таймера MIL
    UINT8 int_timeout_absen;
     //канал
     UINT8 channel;
} 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 11 – Листинг команды

4.1.12 Управление прерываниями по передаче из подадреса - IOCTL_MAN_IRQ_SUBADDRESS_TR

Позволяет управлять прерываниями по передаче из подадреса.

Описание параметров команды приведено на рисунке 12.

```
#define IOCTL_MAN_IRQ_SUBADDRESS_TR
     CTL_CODE(FILE_DEVICE_CONTROLLER, 38, METHOD_BUFFERED,
FILE_ANY_ACCESS)

//входные данные
typedef struct {
    /// \brief состояние прерывания (вкл - 1 / выкл - 2)
    UINT8 rt_int;
    /// \brief маскирование прерываний по подадресам 1й бит - 1й
подадрес и т.д.
    UINT32 subaddress_mask;
     //канал
     UINT8 channel;
} INTERRUPT_SUBADDRESS;

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

Рисунок 12 – Листинг команды

4.1.13 Управление прерываниями по приёму в подадрес - IOCTL_MAN_IRQ_SUBADDRESS_RCV

Позволяет управлять прерываниями по приёму в подадрес.

Описание параметров команды приведено на рисунке 13.

```
#define IOCTL_MAN_IRQ_SUBADDRESS_RCV
     CTL_CODE(FILE_DEVICE_CONTROLLER, 39, METHOD_BUFFERED,
FILE_ANY_ACCESS)

//входные данные
typedef struct {
    /// \brief состояние прерывания (вкл - 1 / выкл - 2)
    UINT8 rt_int;
    /// \brief маскирование прерываний по подадресам 1й бит - 1й
подадрес и т.д.
    UINT32 subaddress_mask;
     //канал
     UINT8 channel;
} INTERRUPT_SUBADDRESS;

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

Рисунок 13 – Листинг команды

4.1.14 Выбор режима работы канала - IOCTL_SWITCH_MODE

Позволяет выбрать режим работы канала.

Описание параметров команды приведено на рисунке 14.

```
#define IOCTL_SWITCH_MODE
     CTL_CODE(FILE_DEVICE_CONTROLLER, 4, METHOD_BUFFERED,
FILE_ANY_ACCESS)

//входные данные
typedef struct _MODE_DATA{
     UINT8 channel; //номер канала
     UINT8 mode;    //режим работы канала
} MODE_DATA, *PMODE_DATA;

/// \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 14 – Листинг команды

4.1.15 Выбор шины канала - IOCTL_SWITCH_BUS

Позволяет выбрать шину канала.

Описание параметров команды приведено на рисунке 15.

```
#define IOCTL_SWITCH_BUS
     CTL_CODE(FILE_DEVICE_CONTROLLER, 33, METHOD_BUFFERED,
FILE_ANY_ACCESS)

//входные данные
typedef struct _MODE_DATA{
     UINT8 channel; //номер канала
     UINT8 mode;    //шина канала
} MODE_DATA, *PMODE_DATA;

/// \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 15 – Листинг команды

4.1.16 Управление работой канала - IOCTL_DEV_CHANNEL_WORK

Позволяет управлять работой канала.

Описание параметров команды приведено на рисунке 16.

```
#define IOCTL_DEV_CHANNEL_WORK
     CTL_CODE(FILE_DEVICE_CONTROLLER, 34, METHOD_BUFFERED,
FILE_ANY_ACCESS)

//входные данные
typedef struct _MODE_DATA{
     UINT8 channel; //номер канала
     UINT8 mode;    //константа включения/выключения канала
} MODE_DATA, *PMODE_DATA;

/// \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 16 – Листинг команды

4.1.17 Установить режим ОУ - IOCTL_SET_RT_TR_MODE

Позволяет установить режим ОУ.

Описание параметров команды приведено на рисунке 17.

```
#define IOCTL_SET_RT_TR_MODE  CTL_CODE(FILE_DEVICE_CONTROLLER, 35,
METHOD_BUFFERED, FILE_ANY_ACCESS)

//входные данные
typedef struct {
    // подадрес
    UINT8 subaddress;
    // режим
    UINT32 mode;
     //канал
     UINT8 channel;
} RT_TR_MODE;

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

Рисунок 17 – Листинг команды

4.1.18 Установить готовность буфера ОУ - IOCTL_SET_RT_TR_BUF_READY

Позволяет установить готовность буфера ОУ.

Описание параметров команды приведено на рисунке 18.

```
#define IOCTL_SET_RT_TR_BUF_READY  CTL_CODE(FILE_DEVICE_CONTROLLER,
37, METHOD_BUFFERED, FILE_ANY_ACCESS)

//входные данные
typedef struct {
    // подадрес 1 - 30
    UINT8 subaddress;
    // команда упр. буфером
    UINT32 cmd_buf;
     //канал
     UINT8 channel;
} 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 18 – Листинг команды

4.1.19 Управление разрешением буферов - IOCTL_RT_BUF_MAN_EN

Позволяет осуществлять управление разрешением буферов.

Описание параметров команды приведено на рисунке 19.

```
#define IOCTL_RT_BUF_MAN_EN   CTL_CODE(FILE_DEVICE_CONTROLLER, 36,
METHOD_BUFFERED, FILE_ANY_ACCESS)

//входные данные
typedef struct {
    // направление (приём/передача)
    UINT32 direction;
    // маскирование (выбор) по подадресам 1й бит - 1й подадрес и
т.д.
    /// 1 - бит изменяем, 0 - бит не изменяем
    UINT32 subaddress_mask;
    // маскирование (выбор) действий по подадресам 1й бит - 1й
подадрес и т.д.
    // 1 - вкл, 0 - выкл
    UINT32 action_mask;
     //канал
     UINT8 channel;
} 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 19 – Листинг команды

4.1.20 Установить адрес ОУ - IOCTL_SET_ADDRESS

Позволяет установить адрес ОУ.

Описание параметров команды приведено на рисунке 20.

```
#define IOCTL_SET_ADDRESS     CTL_CODE(FILE_DEVICE_CONTROLLER, 19,
METHOD_BUFFERED, FILE_ANY_ACCESS)

//входные данные
typedef struct _ADDRESS_REQ{
     UINT8 channel;
     UINT8 address;
} ADDRESS_REQ, *PADDRESS_REQ;
```

Рисунок 20 – Листинг команды

4.1.21 Установить таймауты ОУ - IOCTL_SET_TIMER_RTBM_CONF_REG_PCI

Позволяет установить таймауты ОУ.

Описание параметров команды приведено на рисунке 21.

```
#define IOCTL_SET_TIMER_RTBM_CONF_REG_PCI
     CTL_CODE(FILE_DEVICE_CONTROLLER, 20, METHOD_BUFFERED,
FILE_ANY_ACCESS)

//входные данные
typedef struct {
    UINT32 tck; // значение RT_TCK
    UINT32 trck; // значение RT_TRCK
    UINT32 rcvck; //  значение RT_RCVCK
    UINT8 channel; // канал
} 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 21 – Листинг команды

4.1.22 Установить таймауты КШ - IOCTL_SET_TIMER_BC_CONF_REG_PCI

Позволяет установить таймауты КШ.

Описание параметров команды приведено на рисунке 22.

```
#define IOCTL_SET_TIMER_BC_CONF_REG_PCI
     CTL_CODE(FILE_DEVICE_CONTROLLER, 32, METHOD_BUFFERED,
FILE_ANY_ACCESS)

//входные данные
typedef struct {
    UINT32 tck; // значение RT_TCK
    UINT32 trck; // значение RT_TRCK
    UINT32 rcvck; //  значение RT_RCVCK
    UINT8 channel; // канал
} 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 22 – Листинг команды

4.1.23 Записать подадрес на отправку - IOCTL_WR_BLOCK_BUF_SUBADDR

Позволяет записать данные в подадрес на отправку.

Описание параметров команды приведено на рисунке 23.

```
#define IOCTL_WR_BLOCK_BUF_SUBADDR CTL_CODE(FILE_DEVICE_CONTROLLER,
27, METHOD_BUFFERED, FILE_ANY_ACCESS)

//входные данные
typedef struct {
    UINT8 num_buf;              // номер буфера
    UINT8 subaddress;           // подадрес (1-30)
    UINT16 data_words[32];      // подадрес (32 слова данных)
     UINT8 channel;             // канал
} 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 23 – Листинг команды

4.1.24 Прочитать подадрес на отправку - IOCTL_RD_BLOCK_BUF_SUBADR

Позволяет прочитать данные из подадреса на отправку.

Описание параметров команды приведено на рисунке 24.

```
#define IOCTL_RD_BLOCK_BUF_SUBADDR CTL_CODE(FILE_DEVICE_CONTROLLER,
28, METHOD_BUFFERED, FILE_ANY_ACCESS)

//входные данные
typedef struct {
    UINT8 num_buf;              // номер буфера
    UINT8 subaddress;           // подадрес (1-30)
    UINT16 data_words[32];      // неприменимо
     UINT8 channel;             // канал
} RT_TR_BUF_SUBADDR;

//выходные данные
typedef struct {
    UINT8 num_buf;              // номер буфера
    UINT8 subaddress;           // подадрес (1-30)
    UINT16 data_words[32];      // подадрес (32 слова данных)
     UINT8 channel;             // канал
} 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 24 – Листинг команды

4.1.25 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 25.

```
#define IOCTL_WRITE_BC_RAM    CTL_CODE(FILE_DEVICE_CONTROLLER, 29,
METHOD_BUFFERED, FILE_ANY_ACCESS)

//входные данные
typedef struct {
    /// \brief тип инструкций
    /// INSTR, OPERATION, DATA
    UINT32 type;
    /// \brief смещение от начала буфера в 32-х разрядных словах
    /// INSTR - max 4095, OPERATION - max 4095, DATA - max 8191
    UINT32 shift;
    // размер поля dwords
    UINT32 length;
    // данные
    UINT32 dwords[BC_RAM_MAX];
     // канал
     UINT8 channel;
} 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 25 – Листинг команды

4.1.26 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 26.

```
#define IOCTL_READ_BC_RAM     CTL_CODE(FILE_DEVICE_CONTROLLER, 30,
METHOD_BUFFERED, FILE_ANY_ACCESS)

//входные данные
typedef struct {
    // неприменимо
    UINT32 type;
    /// \brief смещение от начала буфера в 32-х разрядных словах
    /// INSTR - max 4095, OPERATION - max 4095, DATA - max 8191
    UINT32 shift;
    // размер поля dwords
    UINT32 length;
    // неприменимо
    UINT32 dwords[BC_RAM_MAX];
     // канал
     UINT8 channel;
} BC_RAM_BLOCK;

//выходные данные
typedef struct {
    /// \brief тип инструкций
    /// INSTR, OPERATION, DATA
    UINT32 type;
    /// \brief смещение от начала буфера в 32-х разрядных словах
    /// INSTR - max 4095, OPERATION - max 4095, DATA - max 8191
    UINT32 shift;
    // размер поля dwords
    UINT32 length;
    // данные
    UINT32 dwords[BC_RAM_MAX];
     // канал
     UINT8 channel;
} 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 26 – Листинг команды

4.1.27 Получить кол-во блоков прерываний - IOCTL_GET_COUNT_INTERRUPT_BLOCKS

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

Описание параметров команды приведено на рисунке 27.

```
#define IOCTL_GET_COUNT_INTERRUPT_BLOCKS
     CTL_CODE(FILE_DEVICE_CONTROLLER, 50, METHOD_BUFFERED,
FILE_ANY_ACCESS)

//входные данные
UINT8 //номер канала
//выходные данные
UINT32    //количество блоков
```

Рисунок 27 – Листинг команды

4.1.28 Считать блоки прерываний - IOCTL_READ_INTERRUPT_BLOCKS

Позволяет считать заданное кол-во блоков из кольцевого буфера.

Описание параметров команды приведено на рисунке 28.

```
#define IOCTL_READ_INTERRUPT_BLOCKS   CTL_CODE(FILE_DEVICE_CONTROLLER, 51,
METHOD_BUFFERED, FILE_ANY_ACCESS)

struct block_info {
    UINT32    free_timer_value;  // значение таймера в момент
прерывания
    UINT32    interrupt_ch; // маска прерываний
    UINT32    hw_stat_reg1; // регистр HW_STAT_REG1
    UINT32    hw_stat_reg2; // регистр HW_STAT_REG2
};

#define INTERRUPT_BLOCKS_MAX 2048

// входные данные
//блок данных для чтения накопленных прерываний
typedef struct {
    UINT8          channel;
    UINT32         count_blocks;
    // неприменимо
    struct block_info  blocks[INTERRUPT_BLOCKS_MAX];
} INTERRUPT_BLOCK_BUFFER;

//выходные данные
//блок данных для чтения накопленных прерываний
typedef struct {
    // кол-во блоков в ответе
    UINT32         count_blocks;
    // массив блоков
    struct block_info  blocks[INTERRUPT_BLOCKS_MAX];
} INTERRUPT_BLOCK_BUFFER;
// биты маски прерываний
#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 28 – Листинг команды

4.1.29 Считать входной подадрес - IOCTL_GET_RCV_SUBADR_DATA

Позволяет считать данные из входного подареса.

Описание параметров команды приведено на рисунке 29.

```
#define IOCTL_GET_RCV_SUBADDR_DATA
     CTL_CODE(FILE_DEVICE_CONTROLLER, 52, METHOD_BUFFERED,
FILE_ANY_ACCESS)

//входные данные
typedef struct {
    /// \brief подадрес (1-30)
    UINT8 sa;
    //неприменимо
    UINT16 words[32];
    // канал
    UINT8 channel;
} SUBA_DATA_BLOCK;

//выходные данные
typedef struct {
    //  подадрес (1-30)
    UINT8 sa;
    //  слова данных
    UINT16 words[32];
    // канал
    UINT8 channel;
} SUBA_DATA_BLOCK;
```

Рисунок 29 – Листинг команды

4.2 ВЗАИМОДЕЙСТВИЕ С ДРАЙВЕРОМ MIL1553UD

Взаимодействие с драйвером производится с помощью IOCTL вызовов. Определения кодов IOCTL вызовов находятся в файле ioctl_def.h.

Для работы необходимо сперва получить HANDLE устройства с помощью CreateFile:

```
HANDLE getDevice(){
	HANDLE hDevice = NULL;

	hDevice = CreateFile(L"\\\\?\\GLOBALROOT\\Device\\mil1553ud_0",
		GENERIC_READ | GENERIC_WRITE,
		FILE_SHARE_READ | FILE_SHARE_WRITE,
		NULL,
		OPEN_EXISTING,
		0,
		NULL);
	if(hDevice == INVALID_HANDLE_VALUE){
		printf("Error: x%X\n", GetLastError());
	}
	return hDevice;
}
```

Рисунок 30 – листинг получения HANDLE устройства

4.3 Пример вызова IOCTL

После успешного открытия устройства следует осуществить IOCTL вызов VERSION для запроса информации об устройстве. Запрос вернёт коды производителя и устройства, ревизию и количество доступных каналов:

```
     VERSION v;
     HANDLE hDevice = getDevice();
     DWORD ioctl = IOCTL_VERSION, rsize;

     DeviceIoControl(dev, ioctl, NULL, 0, &v, sizeof(VERSION),
&rsize, NULL);

     printf("VEN x%X, DEV x%X rev.%d\tchannels: %d\n", v.vendor_id,
v.device_id, v.revision, v.type);
```

Рисунок 31 – запрос версии устройства

СПИСОК СОКРАЩЕНИЙ

ПО – программное обеспечение;

МКИО – интерфейс по ГОСТ 52070;

КШ – контроллер шины;

ОУ – оконечное устройство;

МШ – монитор шины;

МША– монитор шины адресный;

Лист регистрации

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