УТВЕРЖДАЮ

Генеральный директор

ООО «НОВОМАР»

_____ Т.В. Буга

«____»_____2020 г.

# ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ «ДРАЙВЕР MIL1553UD» И «БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ MIL1553UD»

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

**(ОС LINUX)**

Руководство программиста

ЛИСТ УТВЕРЖДЕНИЯ

ХХХ-ЛУ

| | |
|---|---|
| От | Инженер-программист |
| _____ | _____ _____ |
| «____»_____2020 г. | «____»_____2020 г. |

2020

| | | |
|---|---|---|
| | | |
| | | |
| *Из* | *Под* | *Дат* |

Литера

| | |
|---|---|
| *Подп. и* | |
| *Инв. № дубл* | |
| *Взам. инв. №* | |
| *Подп. и* | |
| *Инв. № подл* | |

Утвержден

ЛУ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«ДРАЙВЕР MIL1553UD» И «БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ MIL1553UD»

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

**(ОС LINUX)**

Руководство программиста

ХХХ

Листов 53

2020

| 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» И «БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ MIL1553UD» (ОС Linux и Astra Linux), 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 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ MIL1553UD

Программное обеспечение «БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ MIL1553UD» (далее – библиотека) обеспечивает вспомогательный сервисный функционал при взаимодействии с PCI-устройством MIL1553UD.

Библиотека обеспечивает выполнение следующих основных задач:

– реализация сервисных функций поканально.

# 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, инструкция по сборке и настройке приведена в приложении А.

## 2.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ MIL1553UD

Библиотека предназначена для функционирования в ОС Linux, Astra Linux и встраивания в прикладное ПО, как исходный код.

# 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 pci-платы, M — порядковый номер канала на pci-плате.

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

Взаимодействие с каналами происходит посредством ioctl-команд.

### 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 следующий 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

## 3.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ MIL1553UD

Библиотека взаимодействия разработана на языке С и состоит из следующих файлов:

- mil1553ud_library.h;

- mil1553ud_library.c;

- mil1553ud_regs.h;

- mil1553ud_cmd.h.

Для использования библиотеки 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 «mil1553ud_cmd.h», реализация функций ioctl-команд представлена в файле «mil1553ud_ioctl.c».

Пример исходного кода программы-примера приведён в приложении Б.

Скриншоты работы тестовой программы приведены в приложении В.

### 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 информации о pci-локации платы - IOCTL_PCI_LOCATION

Позволяет прочитать информацию о pci-локации платы.

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 из DMA - IOCTL_READ_BLOCKS_RAW_DMA

Позволяет получить данные (блоки) из буфера DMA.

Описание параметров команды приведено на рисунке 8.

```
#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 – Листинг команды

### 4.1.12 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

4.1.13 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

4.1.14 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

4.1.15 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

### 4.1.16 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

### 4.1.17 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

### 4.1.18 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

4.1.19 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

### 4.1.20 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

### 4.1.21 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

### 4.1.22 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

### 4.1.23 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

4.1.24 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

### 4.1.25 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

### 4.1.26 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

### 4.1.27 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

### 4.1.28 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

### 4.1.29 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

### 4.1.30 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

### 4.1.31 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

## 4.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ MIL1553UD

Библиотека предназначена для функционирования в ОС Linux, Astra Linux и встраивания в прикладное ПО, как исходный код. В файле «mil1553ud_library.h» представлены сервисные функции библиотеки взаимодействия.

Пример исходного кода программы-примера приведён в приложении Б.

Скриншоты работы тестовой программы приведены в приложении В.

### 4.2.1 Получить номер канала из символьного устройства канала

Описание функции приведено на рисунке 29.

```
///
/// \brief mil1553ud_getNumberChannel получить номер канала из
символьного утсройства канала
/// \param fd дескриптор файла
/// \return номер канала
///
unsigned int mil1553ud_getNumberChannel(int fd);
```

Рисунок 29 – Листинг функции

### 4.2.2 Генерация имя регистра в соответствии с номером канала

Описание функции приведено на рисунке 30.

```
///
/// \brief макрос генериует имя регистра в соответствии с номером
канала
/// \param numCh номер канала
/// \param RegName имя регистра без номера канала
///
/// \code
/// unsigned int numCh = mil1553ud_getNumberChannel(fd);
/// unsigned int regAddress = MIL_REG(numCh, CTRL_REG_PCI_CH);
/// \endcode
///
#define MIL_REG(numCh, RegName) \
    ( (numCh == CH_NUM_1) ? RegName##1 : ( (numCh == CH_NUM_2) ?
RegName##2 : ( (numCh == CH_NUM_3) ? RegName##3 : RegName##4) ) )
```

Рисунок 30 – Листинг функции

### 4.2.3 Получить количество блоков DMA, готовых для чтения

Описание функции приведено на рисунке 31.

```
///
/// \brief mil1553ud_getCountDmaBlocksReadyRead получить количество
блоков DMA, готовых для чтения
/// \param fd дескриптор файла
/// \return кол-во блоков DMA, готовых к чтению
///
unsigned int mil1553ud_getCountDmaBlocksReadyRead(int fd);
```

Рисунок 31 – Листинг функции

### 4.2.4 Создать контейнер блоков DMA

Описание функции приведено на рисунке 32.

```
///
/// \brief mil1553ud_newDmaReadBlock создать контейнер блоков DMA
/// выделяет под контейнер память
/// \param countDmaBlocks размер контейнера в блоках (штуках)
/// \return указатель на контейнер
///
DMA_READ_BLOCK* mil1553ud_newDmaReadBlock(int countDmaBlocks);
```

Рисунок 32 – Листинг функции

### 4.2.5 Освобождает контейнер блоков DMA

Описание функции приведено на рисунке 33.

```
///
/// \brief mil1553ud_freeDmaReadBlock освобождает контейнер блоков
DMA
/// \param block указатель на контейнер
///
void mil1553ud_freeDmaReadBlock(DMA_READ_BLOCK* block);
```

Рисунок 33 – Листинг функции

### 4.2.6 Считывает все блоки DMA, готовые к чтению, из канала

Описание функции приведено на рисунке 34.

```
///
/// \brief mil1553ud_readAllDmaBlocksReadyRead считывает все блоки
DMA, готовые к чтению, из канала
/// выделяет под контейнер память
/// \param fd дескриптор файла
/// \return указатель на контейнер
///
DMA_READ_BLOCK* mil1553ud_readAllDmaBlocksReadyRead(int fd);
```

Рисунок 34 – Листинг функции

### 4.2.7 Получить количество блоков прерываний, готовых для чтения

Описание функции приведено на рисунке 35.

```
///
/// \brief mil1553ud_getCountInterruptBlocksReadyRead получить
количество блоков прерываний, готовых для чтения
/// \param fd дескриптор файла
/// \return кол-во блоков, готовых к чтению
///
unsigned int mil1553ud_getCountInterruptBlocksReadyRead(int fd);
```

Рисунок 35 – Листинг функции

### 4.2.8 Создать контейнер блоков прерываний

Описание функции приведено на рисунке 36.

```
///
/// \brief mil1553ud_newInterruptBlock создать контейнер блоков
прерываний
/// \param countBlocks размер контейнера в блоках (штуках)
/// \return указатель на контейнер
///
INTERRUPT_BLOCK_BUFFER* mil1553ud_newInterruptBlock(int
countBlocks);
```

Рисунок 36 – Листинг функции

### 4.2.9 Освобождает контейнер блоков блоков прерываний

Описание функции приведено на рисунке 37.

```
///
/// \brief mil1553ud_freeInterruptBlock освобождает контейнер блоков
блоков прерываний
/// \param block указатель на контейнер
///
void mil1553ud_freeInterruptBlock(INTERRUPT_BLOCK_BUFFER* block);
```

Рисунок 37 – Листинг функции

### 4.2.10 Считывает все блоки прерываний, готовые к чтению, из канала

Описание функции приведено на рисунке 38.

```
///
/// \brief mil1553ud_readAllInterruptBlocksReadyRead считывает все
блоки прерываний, готовые к чтению, из канала
/// \param fd дескриптор файла
/// \return указатель на контейнер
///
INTERRUPT_BLOCK_BUFFER*
mil1553ud_readAllInterruptBlocksReadyRead(int fd);
```

Рисунок 38 – Листинг функции

### 4.2.11 Управление микропрограммой в режиме КШ

Описание функции приведено на рисунке 39.

```
///
/// \brief mil1553ud_bcProgram управление микропрограммой в режиме
КШ
/// старт/стоп BCSTRT
/// \param fd дескриптор файла
/// \param action действие
///
void mil1553ud_bcProgram(int fd, unsigned int action);

/// \brief действие - останов программы
#define BC_PROGRAM_STOP     0x0
/// \brief действие - старт программ
#define BC_PROGRAM_START    ~BC_PROGRAM_STOP
```

Рисунок 39 – Листинг функции

4.2.12 Установить стартовый адрес инструкции микропрограммы КШ

Описание функции приведено на рисунке 40.

```
///
/// \brief mil1553ud_bcSetStartInstructionAddress установить
стартовый адрес интсрукции микропрограммы КШ
/// \param fd дескриптор файла
/// \param startNumber стартовый номер (адрес)
///
void mil1553ud_bcSetStartInstructionAddress(int fd, unsigned int
startNumber);
```

Рисунок 40 – Листинг функции

4.2.13 Создать контейнер для микропрограммы КШ

Описание функции приведено на рисунке 41.

```
///
/// \brief mil1553ud_newBcRamBlock создать контейнер для
микропрограммы КШ
/// \param type тип - инструкции, операции, данные
/// MIL_BC_RAM_TYPE_INSTRUCTION, MIL_BC_RAM_TYPE_OPERATION,
MIL_BC_RAM_TYPE_DATA
/// \param sizeInDwords размер контейнера в 32-разядных (двойных)
словах
/// \param shiftInDwords смещение в памяти КШ в 32-разядных
(двойных) словах
/// \return указатель на контейнер для микропрограммы КШ
///
BC_RAM_BLOCK* mil1553ud_newBcRamBlock(unsigned int type, unsigned
int sizeInDwords, unsigned int shiftInDwords);
```

Рисунок 41 – Листинг функции

4.2.14 Освободить контейнер для микропрограммы КШ

Описание функции приведено на рисунке 42.

```
///
/// \brief mil1553ud_freeBcRamBlock освободить контейнер для
микропрограммы КШ
/// \param block указатель на контейнер для микропрограммы КШ
///
void mil1553ud_freeBcRamBlock(BC_RAM_BLOCK* block);
```

Рисунок 42 – Листинг функции

### 4.2.15 Формирование инструкции для микропрограммы КШ

Описание функции приведено на рисунке 43.

```
///
/// \brief формирование инструкции (32-разрядное слово)
/// \param instruction инструкция
/// \param condition условие
/// \param parameter параметр
///
/// \code
/// unsigned int dword = INSTRUCTION(XEQ, ALWAYS, 0);
/// \endcode
///
#define INSTRUCTION(instruction, condition, parameter)    \
    (   \
    ( 0 << 31 ) | ( (instruction & 0x1F) << 26 ) | ( IDENT << 21 ) |
( (condition & 0x1F) << 16) | (parameter & 0xFFFF) \
    )
```

Рисунок 43 – Листинг функции

### 4.2.16 Формирование нечётной части операции для микропрограммы КШ

Описание функции приведено на рисунке 44.

```
///
/// \brief нечётная часть операции (32-разрядное слово)
/// \param command команда КШ
/// \param timeout время до начала следующей операции
///
#define OPERATION_ODD_PART(command, timeout)     \
    (   \
    ( (command & 0xFFFF) << 16 ) | (timeout & 0xFFFF)   \
    )
```

Рисунок 44 – Листинг функции

### 4.2.17 Формирование чётной части операции для микропрограммы КШ

Описание функции приведено на рисунке 45.

```
///
/// \brief чётная часть операции (32-разрядное слово)
/// \param cw1 КС1
/// \param parameter параметр
///
#define OPERATION_EVEN_PART(cw1, parameter) \
    (   \
    ( (cw1 & 0xFFFF) << 16 ) | (parameter & 0xFFFF)   \
    )
```

Рисунок 45 – Листинг функции

4.2.18 Формирование командного слова операции для микропрограммы КШ

Описание функции приведено на рисунке 46.

```
///
/// \brief командное слово (ГОСТ 52070)
/// \param adrOu адрес ОУ
/// \param k разряд "приём/передача"
/// 0 - ОУ должен принимать СД
/// 1 - ОУ должен передавать СД
/// \param suba_mode подадрес или режим управления
/// \param cntDw_code число СД или код команды
///
#define COMMAND_WORD(adrOu, k, suba_mode, cntDw_code)   \
    (   \
    ( (adrOu & 0x1F) << 11 ) | ( (k & 0x1) << 10) | ( (suba_mode &
0x1F) << 5) | ( (cntDw_code & 0x1F) << 0)    \
    )
```

Рисунок 46 – Листинг функции

4.2.19 Получить указатель на блок DMA из контейнера

Описание функции приведено на рисунке 47.

```
/// \brief получить указатель на блок DMA
#define GET_PTR_DMA_BLOCK(data, numberBlock)  \
    data + MIL_DMA_BLOCK_SIZE * numberBlock
```

Рисунок 47 – Листинг функции

4.2.20 Проверка на допустимость контейнера DMA

Описание функции приведено на рисунке 48.

```
/// \brief проверка на допустимость
#define ASSERT_DMA_CONTAINER(ptrContainerDma, numberBlock)  \
    (ptrContainerDma == NULL) ? MIL1553UD_FALSE :  \
        (numberBlock > ptrContainerDma->countBlocks) ?
MIL1553UD_FALSE : MIL1553UD_TRUE
```

Рисунок 48 – Листинг функции

4.2.21 Получить тип транзакции из блока DMA

Описание функции приведено на рисунке 49.

```
///
/// \brief mil1553ud_dmaBlock_getTypeTransaction получить тип
транзакции из блока DMA
/// тип транзакции соттветствует значениям 1-10 (см. ГОСТ 52070)
/// \param dmaBlock тип блока DMA
/// \param dmaBlock указатель на блок DMA
/// \return тип транзакции
///
unsigned int mil1553ud_dmaBlock_getTypeTransaction(unsigned int
typeDmaBlock, unsigned char* dmaBlock);
```

Рисунок 49 – Листинг функции

4.2.22 Проверить успешность транзакции

Описание функции приведено на рисунке 50.

```
///
/// \brief mil1553ud_dmaBlock_getFreeTimer получить значение
freetimer блока DMA
/// \param typeDmaBlock тип блока DMA
/// \param dmaBlock указатель на блок DMA
/// \return значение freetimer
///
unsigned int mil1553ud_dmaBlock_getFreeTimer(unsigned int
typeDmaBlock, unsigned char* dmaBlock);
```

Рисунок 50 – Листинг функции

4.2.23 Получить количество служебных 16-разрядных слов

Описание функции приведено на рисунке 51.

```
///
/// \brief mil1553ud_dmaBlock_getCountServiceWords16 получить
количество служебных 16-разрядных слов
/// (включая dword 1-5 и служебные слова МКИО)
/// \param typeDmaBlock тип блока DMA
/// \param dmaBlock указатель на блок DMA
/// \return количество служебных 16-разрядных слов
///
unsigned int mil1553ud_dmaBlock_getCountServiceWords16(unsigned int
typeDmaBlock, unsigned char* dmaBlock);
```

Рисунок 51 – Листинг функции

4.2.24 Получить количество 16-разрядных слов данных

Описание функции приведено на рисунке 52.

```
///
/// \brief mil1553ud_dmaBlock_getCountDataWords16 получить
количество 16-разрядных слов данных
/// \param typeDmaBlock тип блока DMA
/// \param dmaBlock указатель на блок DMA
/// \return количество 16-разрядных слов данных
///
unsigned int mil1553ud_dmaBlock_getCountDataWords16(unsigned int
typeDmaBlock, unsigned char* dmaBlock);
```

Рисунок 52 – Листинг функции

4.2.25 Получить активную шину из блока DMA

Описание функции приведено на рисунке 53.

```
///
/// \brief mil1553ud_dmaBlock_getActiveBus получить активную шину
/// (по которой была транзакция)
/// \param typeDmaBlock тип блока DMA
/// \param dmaBlock указатель на блок DMA
/// \return шина А или Б
/// MIL1553_BUS_A или MIL1553_BUS_B
///
unsigned int mil1553ud_dmaBlock_getActiveBus(unsigned int
typeDmaBlock, unsigned char* dmaBlock);

/// \brief Шина "А"
#define MIL1553_BUS_A       0x0
/// \brief Шина "Б"
#define MIL1553_BUS_B       ~MIL1553_BUS_A
```

Рисунок 53 – Листинг функции

4.2.26 Получить указатель на часть с данными блока DMA

Описание функции приведено на рисунке 54.

```
///
/// \brief mil1553ud_dmaBlock_getPtrDataPart получить указатель на
часть с данными блока DMA
/// \param dmaBlock указатель на блок DMA
/// \return указатель на часть с данными блока DMA
///
unsigned char* mil1553ud_dmaBlock_getPtrDataPart(unsigned char*
dmaBlock);
```

Рисунок 54 – Листинг функции

4.2.27 Получить подадрес из блока DMA в режиме ОУ

Описание функции приведено на рисунке 55.

```
///
/// \brief mil1553ud_dmaBlock_tdModeSubaddress получить поддарес из
блока DMA в режиме ОУ
/// \param dmaBlock указатель на блок DMA
/// \return поддарес
///
unsigned int mil1553ud_dmaBlock_tdModeSubaddress(unsigned char*
dmaBlock);
```

Рисунок 55 – Листинг функции

# 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 56.

```
/// \brief нет ошибок
#define GOOD                            0
/// \brief ошибка
#define ERROR                           -1000
/// \brief ошибка адреса регистра
#define ERR_BAD_ADDRESS                 -1001
/// \brief ошибка номера канала
/// внутрення 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 56 – Листинг

## 5.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ MIL1553UD

Описание кодов ошибок приведено на рисунке 57.

```
/// \brief невалидное значение, свидетельствующее об ошибке
#define LIB_BAD_VALUE       0xFFFFFFFF
/// \brief значение отсутсвия ошибки
#define LIB_GOOD            0
```

Рисунок 57 - Листинг

ПРИЛОЖЕНИЕ А (ОБЯЗАТЕЛЬНОЕ)

Инструкция по сборке и установке драйвера

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

- с помощью командной строки и утилиты make:

открыть терминал в папке с проектом, написать в терминале "make"

для очистки проекта - "make clean"

для установки драйвера - "sudo make install"

для удаления драйвера - "sudo make uninstall"

для останова работающего драйвера - "sudo rmmod mil1553ud_driver"

для запуска установленного драйвера - "sudo insmod mil1553ud_driver.ko"

для проверки работает ли драйвер в данный момент - "sudo lsmod | grep mil1553ud_driver"

ТАК ЖЕ перед началом работ следует установить:

"sudo apt-get install libelf-dev"

"sudo apt-get install linux-headers-generic".

ПРИЛОЖЕНИЕ Б (ИНФОРМАЦИОННОЕ)

# Листинг программы-примера

Рисунок Б1 – Листинг инициализации программных таймеров

```
    channelTimerBC = new QTimer(this);
    connect(channelTimerBC, &QTimer::timeout, this, &MainWindow::slot_timerDMA);
    channelTimerRT = new QTimer(this);
    connect(channelTimerRT, &QTimer::timeout, this,
&MainWindow::slot_timerDMA_RT);

    connect(ui->pushButton_initBC, &QPushButton::clicked, this,
&MainWindow::slot_initBC);
  connect(ui->pushButton_initRT, &QPushButton::clicked, this, &MainWindow::slot_initRT);
```

Рисунок Б2 – Листинг инициализации КШ

```
    auto switchOffChannel = [this](void)->void {
        // выключаем канал
        unsigned int chOn = MIL_DEV_CHANNEL_OFF;
        ioctl(this->chDevice, IOCTL_DEV_CHANNEL_WORK, &chOn);
        // сбрасываем DMA
        ioctl(this->chDevice, IOCTL_DISABLE_DMA);
        ioctl(this->chDevice, IOCTL_CLEAR_DMA);
    };
    auto initChannel = [this](void)->void {
        // включаем прерывания
        INTERRUPT_MAN iman;
        iman.int_bc = INTERRUPT_MAN_ON;
        ioctl(this->chDevice, IOCTL_MANAGE_INTERRUPT, &iman);
        // выбор режима
        unsigned int modeBus = MIL_MODE_BUS_CONTR; // режим КШ
        ioctl(this->chDevice, IOCTL_SWITCH_MODE, &modeBus);
        // выбор шины
        unsigned int bus = MIL_BUS_A_EN | MIL_BUS_B_EN; // шины А и Б
        ioctl(this->chDevice, IOCTL_SWITCH_BUS, &bus);
    };
    auto readRegister = [this](unsigned long regAdr, QString name)->void {
        SADDR_DATA dataAdr;     // регистр
        dataAdr.daddr = regAdr;
        ioctl(this->chDevice, IOCTL_READ_REG, &dataAdr);
        emit signal_writeInTerminal(QString("%1 = %2b")
                                    .arg(name)
                                    .arg(QString::number(dataAdr.data, 2))
                                    );
    };
    auto writeMicroProgram = [this](void)->void {
        // составляем микропрограмму
        BC_RAM_BLOCK* block_instruction;
        // формируем инструкции
        block_instruction = mil1553ud_newBcRamBlock(MIL_BC_RAM_TYPE_INSTRUCTION,
4, 0);
        block_instruction->dwords[0] = INSTRUCTION(XEQ, ALWAYS, 0);
        block_instruction->dwords[1] = INSTRUCTION(XEQ, ALWAYS, 2);
        block_instruction->dwords[2] = INSTRUCTION(IRQ, 1, 0);
        block_instruction->dwords[3] = INSTRUCTION(HLT, ALWAYS, 0);
        // формируем операции
        BC_RAM_BLOCK* block_operation;
        block_operation = mil1553ud_newBcRamBlock(MIL_BC_RAM_TYPE_OPERATION, 4,
```

```
0);
        unsigned int adr = ui->spinBox->value();
        emit signal_writeInTerminal(QString("adr =
%1").arg(QString::number(adr)));
        block_operation->dwords[0] = OPERATION_ODD_PART(0x1, 0);
        block_operation->dwords[1] = OPERATION_EVEN_PART( COMMAND_WORD(adr, 0, 1,
16), 0);
        block_operation->dwords[2] = OPERATION_ODD_PART(0x2, 0);
        block_operation->dwords[3] = OPERATION_EVEN_PART( COMMAND_WORD(adr, 1, 2,
2), 0);
        // формируем данные
        BC_RAM_BLOCK* block_data;
        block_data = mil1553ud_newBcRamBlock(MIL_BC_RAM_TYPE_DATA, 2, 0);
        block_data->dwords[0] = (0xAAAA<<16) | 0x5555;
        block_data->dwords[1] = (0xAAAA<<16) | 0x5555;
        // записываем микропрограмму
        ioctl(this->chDevice, IOCTL_WRITE_BC_RAM, block_instruction);
        ioctl(this->chDevice, IOCTL_WRITE_BC_RAM, block_operation);
        ioctl(this->chDevice, IOCTL_WRITE_BC_RAM, block_data);
        // освобождаем буферы микропрограммы
        mil1553ud_freeBcRamBlock(block_instruction);
        mil1553ud_freeBcRamBlock(block_operation);
        mil1553ud_freeBcRamBlock(block_data);
    };
    auto switchOnChannel = [this](unsigned int numChannel)->void {
        Q_UNUSED(numChannel)
        // включаем канал
        unsigned int chOn = MIL_DEV_CHANNEL_ON;
        ioctl(this->chDevice, IOCTL_DEV_CHANNEL_WORK, &chOn);
        // включаем DMA
        ioctl(this->chDevice, IOCTL_ENABLE_DMA);
        mil1553ud_bcSetStartInstructionAddress(this->chDevice, 0); // стартовая
инструкция микропрограммы
    };

    // режим КШ
    this->chDevice = open(QString("/dev/mil1553dev-0-ch-
0").toLatin1().constData(), O_RDWR);

    if (this->chDevice <= 0) {
        emit signal_writeInTerminal("Символьное устройство канала не найдено!");
        return;
    }

    unsigned int numChannel = mil1553ud_getNumberChannel(this->chDevice);
    emit signal_writeInTerminal(QString("Канал =
%1").arg(QString::number(numChannel+1)));

    switchOffChannel();
    readRegister(MIL_REG(numChannel, CTRL_REG_PCI_CH), "CTRL_REG_PCI_CH"); //
читаем регистр CTRL_REG_PCI_CH
    initChannel();
    writeMicroProgram();
    switchOnChannel(numChannel);
    mil1553ud_bcProgram(this->chDevice, BC_PROGRAM_START); // BCSTRT - старт
выполнения программы КШ
    readRegister(MIL_REG(numChannel, CTRL_REG_PCI_CH), "CTRL_REG_PCI_CH"); //
читаем регистр CTRL_REG_PCI_CH

    this->chMode = MIL_MODE_BUS_CONTR;
```

```
channelTimerBC->start(2000);   // старт опросного таймер
```

Рисунок Б3 – Листинг инициализации ОУ

```
    auto switchOffChannel = [this](void)->void {
        // вЫключаем канал
        unsigned int chOn = MIL_DEV_CHANNEL_OFF;
        ioctl(this->chDevice, IOCTL_DEV_CHANNEL_WORK, &chOn);
        ioctl(this->chDevice, IOCTL_DISABLE_DMA);
        ioctl(this->chDevice, IOCTL_CLEAR_DMA);
    };
    auto initChannel = [this](void)->void {
        unsigned int modeBus = MIL_MODE_TERMINAL_DEV;
        unsigned int bus = MIL_BUS_A_EN | MIL_BUS_B_EN;
        unsigned char address = 2;
        ioctl(this->chDevice, IOCTL_SWITCH_MODE, &modeBus);
        ioctl(this->chDevice, IOCTL_SET_ADDRESS, &address);
        ioctl(this->chDevice, IOCTL_SWITCH_BUS, &bus);
        emit signal_writeInTerminal(QString("Адрес =
%1").arg(QString::number(address)));
    };
    auto initSubaddresses = [this](void)->void {
        // на приём
        RT_BUF_MAN_EN rcvEn;
        rcvEn.direction = MIL_RT_BUF_RECEIVE;
        rcvEn.subaddress_mask = 0x7FFFFFFE; // меняем с 1 по 30 подадрес
        rcvEn.action_mask = 0x7FFFFFFE; // включаем на приём с 1 по 30 подадрес
        ioctl(this->chDevice, IOCTL_RT_BUF_MAN_EN, &rcvEn);
        // на передачу
        RT_BUF_MAN_EN trnEn;
        trnEn.direction = MIL_RT_BUF_TRANSMIT;
        trnEn.subaddress_mask = 0x7FFFFFFE; // меняем с 1 по 30 подадрес
        trnEn.action_mask = 0x7FFFFFFE; // включаем на приём с 1 по 30 подадрес
        ioctl(this->chDevice, IOCTL_RT_BUF_MAN_EN, &trnEn);
    };
    auto initSubaddressBuffer = [this](void)->void {
        for (int i = 1; i <= 30; i++) {
            RT_TR_MODE rt_tr_mode;
            rt_tr_mode.mode = MIL_RT_MODE_PROG;
            rt_tr_mode.subaddress = i;
            ioctl(this->chDevice, IOCTL_SET_RT_TR_MODE, &rt_tr_mode);
            RT_TR_BUF_SUBADDR bufSub;
            bufSub.num_buf = MIL_RT_BUF0;
            bufSub.subaddress = i;
            bufSub.data_words[0] = 0x1111;
            bufSub.data_words[1] = 0x2222;
            ioctl(this->chDevice, IOCTL_WR_BLOCK_BUF_SUBADR, &bufSub);

            RT_TR_MODE_BUF rdBuf;
            rdBuf.cmd_buf = MIL_RT_BUF_0_ON_1_OFF;
            rdBuf.subaddress = i;
            ioctl(this->chDevice, IOCTL_SET_RT_TR_BUF_READY, &rdBuf);
        }
    };
    auto initInterrupt = [this](unsigned int numChannel)->void {
        // включаем прерывания
        INTERRUPT_MAN iman;
```

```
        iman.int_rt_ren = INTERRUPT_MAN_ON;
        iman.int_rt_ten = INTERRUPT_MAN_ON;
        ioctl(this->chDevice, IOCTL_MANAGE_INTERRUPT, &iman);

        INTERRUPT_SUBADDRESS intSub;
        intSub.rt_int = INTERRUPT_MAN_ON;
        intSub.subaddress_mask = 0x7FFFFFFE;
        ioctl(this->chDevice, IOCTL_MAN_IRQ_SUBADDRESS_TR, &intSub);

        intSub.rt_int = INTERRUPT_MAN_ON;
        intSub.subaddress_mask = 0x7FFFFFFE;
        ioctl(this->chDevice, IOCTL_MAN_IRQ_SUBADDRESS_RCV, &intSub);

        SADDR_DATA_BIT_MASK bitMask;
        bitMask.daddr = MIL_REG(numChannel, RT_INT_REG_CH);
        bitMask.mask = (1<<0) | (1<<1) | (1<<2);
        bitMask.data = (1<<0) | (1<<1) | (1<<2);
        ioctl(this->chDevice, IOCTL_WRITE_REG_BIT_MASK, &bitMask);
    };
    auto switchOnChannel = [this](void)->void {
        ioctl(this->chDevice, IOCTL_ENABLE_DMA);
        // включаем канал
        unsigned int chOn = MIL_DEV_CHANNEL_ON;
        ioctl(this->chDevice, IOCTL_DEV_CHANNEL_WORK, &chOn);
    };

    // режим ОУ
    this->chDevice = open(QString("/dev/mil1553dev-0-ch-
2").toLatin1().constData(), O_RDWR);

    if (this->chDevice <= 0) {
        emit signal_writeInTerminal("Символьное устройство канала не найдено!");
        return;
    }

    unsigned int numChannel = mil1553ud_getNumberChannel(this->chDevice);
    emit signal_writeInTerminal(QString("Канал =
%1").arg(QString::number(numChannel+1)));

    switchOffChannel();
    initChannel();
    initSubaddresses();
    initSubaddressBuffer();
    initInterrupt(numChannel);
    switchOnChannel();

    this->chMode = MIL_MODE_TERMINAL_DEV;
  channelTimerRT->start(2000); // старт опросного таймера
```

Рисунок Б4 – Листинг разбора DMA КШ

```
void MainWindow::slot_timerDMA()
{
    unsigned int countDmaBlocks = mil1553ud_getCountDmaBlocksReadyRead(this-
>chDevice); // чтение кол-во блоков DMA для чтения
    unsigned int countInterrupts =
mil1553ud_getCountInterruptBlocksReadyRead(this->chDevice); // чтения кол-во
блоков прерываний для четния
    if (countInterrupts) {
        emit signal_writeInTerminal(QString("countInterrupts =
%1").arg(QString::number(countInterrupts)));
        INTERRUPT_BLOCK_BUFFER* intBlock =
mil1553ud_readAllInterruptBlocksReadyRead(this->chDevice);
        for (unsigned int i = 0; i < intBlock->count_blocks; i++) {
            emit signal_writeInTerminal(QString("freetimer = %1, interrupt_ch =
%2b")
                                        .arg(QString::number(intBlock-
>blocks[i].free_timer_value))
                                        .arg(QString::number(intBlock-
>blocks[i].interrupt_ch, 2))
                                        );
        }
        mil1553ud_freeInterruptBlock(intBlock);
    }
    if (countDmaBlocks == 0)
        return;
    emit signal_writeInTerminal("timerDMA");
    DMA_READ_BLOCK* dmaBlockContainer = mil1553ud_readAllDmaBlocksReadyRead(this-
>chDevice); // считываем все блоки DMA готовые для чтения
    if (dmaBlockContainer == NULL) {
        emit signal_writeInTerminal("dmaBlock == NULL");
        return;
    }
    for (unsigned int indexBlock = 0; indexBlock < dmaBlockContainer->countBlocks;
indexBlock++) {
        if (ASSERT_DMA_CONTAINER(dmaBlockContainer, indexBlock) ==
MIL1553UD_FALSE)
            continue;
        unsigned char* block = GET_PTR_DMA_BLOCK(dmaBlockContainer->data,
indexBlock); // текущий 128 байтный блок DMA
        processingDmaBlockBc(block);// разбор блока DMA КШ
    }
    mil1553ud_freeDmaReadBlock(dmaBlockContainer);
}


void MainWindow::processingDmaBlockBc(unsigned char *block)
{
    unsigned int typeDmaBlock = MIL1553UD_BC_DMA_BLOCK; // КШ
    unsigned int isSuccessTransaction =
mil1553ud_dmaBlock_isSuccessTransaction(typeDmaBlock, block);
    unsigned int typeTransaction =
mil1553ud_dmaBlock_getTypeTransaction(typeDmaBlock, block);
    unsigned int freeTimer = mil1553ud_dmaBlock_getFreeTimer(typeDmaBlock, block);
    unsigned int countServiceWords16 =
mil1553ud_dmaBlock_getCountServiceWords16(typeDmaBlock, block);
    unsigned int countDataWords16 =
mil1553ud_dmaBlock_getCountDataWords16(typeDmaBlock, block);
```

```
    unsigned int activeBus = mil1553ud_dmaBlock_getActiveBus(typeDmaBlock, block);
    emit signal_writeInTerminal(
                QString("Транзакция - тип = %1 - %2 - freeTimer: %3 -
CntServWords: %4 - CntDataWords: %5 - Шина: %7")
                .arg(QString::number(typeTransaction))
                .arg( (isSuccessTransaction) ? "нет ошибок" : "ошибка" )
                .arg(QString::number(freeTimer))
                .arg(QString::number(countServiceWords16))
                .arg(QString::number(countDataWords16))
                .arg( (activeBus == MIL1553_BUS_A) ? "Шина А" : "Шина Б" )
                                );
    if (countDataWords16 > 0) {
        unsigned short* dataWord16 = (unsigned
short*)mil1553ud_dmaBlock_getPtrDataPart(block);
        QString dataWordsString = QString("Слова данных:");
        for(unsigned int i = 0; i < countDataWords16; i++) {
            if (i == countDataWords16-1) {
                dataWordsString += QString(" 0x%1;")
                        .arg(QString::number(dataWord16[i], 16));
            } else {
                dataWordsString += QString(" 0x%1,")
                        .arg(QString::number(dataWord16[i], 16));
            }
        }
        emit signal_writeInTerminal(dataWordsString);
    }
}
```

Рисунок Б5– Листинг разбора DMA ОУ

```
void MainWindow::slot_timerDMA_RT()
{
    unsigned int countDmaBlocks = mil1553ud_getCountDmaBlocksReadyRead(this-
>chDevice); // чтение кол-во блоков DMA для чтения
    unsigned int countInterrupts =
mil1553ud_getCountInterruptBlocksReadyRead(this->chDevice); // чтения кол-во
блоков прерываний для четния
    if (countInterrupts) {
        emit signal_writeInTerminal(QString("countInterrupts =
%1").arg(QString::number(countInterrupts)));
        INTERRUPT_BLOCK_BUFFER* intBlock =
mil1553ud_readAllInterruptBlocksReadyRead(this->chDevice);
        for (unsigned int i = 0; i < intBlock->count_blocks; i++) {
            emit signal_writeInTerminal(QString("freetimer = %1, interrupt_ch =
%2b")
                                        .arg(QString::number(intBlock-
>blocks[i].free_timer_value))
                                        .arg(QString::number(intBlock-
>blocks[i].interrupt_ch, 2))
                                        );
        }
        mil1553ud_freeInterruptBlock(intBlock);
    }
    if (countDmaBlocks == 0)
        return;
    emit signal_writeInTerminal("timerDMA");
    DMA_READ_BLOCK* dmaBlockContainer = mil1553ud_readAllDmaBlocksReadyRead(this-
>chDevice); // считываем все блоки DMA готовые для чтения
    if (dmaBlockContainer == NULL) {
        emit signal_writeInTerminal("dmaBlock == NULL");
        return;
    }
    for (unsigned int indexBlock = 0; indexBlock < dmaBlockContainer->countBlocks;
indexBlock++) {
        if (ASSERT_DMA_CONTAINER(dmaBlockContainer, indexBlock) ==
MIL1553UD_FALSE)
            continue;
        unsigned char* block = GET_PTR_DMA_BLOCK(dmaBlockContainer->data,
indexBlock); // текущий 128 байтный блок DMA
        processingDmaBlockTd(block);// разбор блока DMA ОУ
    }
    mil1553ud_freeDmaReadBlock(dmaBlockContainer);
}


void MainWindow::processingDmaBlockTd(unsigned char *block)
{
    unsigned int typeDmaBlock = MIL1553UD_TD_DMA_BLOCK;     // ОУ
    unsigned int isSuccessTransaction =
mil1553ud_dmaBlock_isSuccessTransaction(typeDmaBlock, block);
    unsigned int typeTransaction =
mil1553ud_dmaBlock_getTypeTransaction(typeDmaBlock, block);
    unsigned int freeTimer = mil1553ud_dmaBlock_getFreeTimer(typeDmaBlock, block);
    unsigned int countServiceWords16 =
mil1553ud_dmaBlock_getCountServiceWords16(typeDmaBlock, block);
    unsigned int countDataWords16 =
mil1553ud_dmaBlock_getCountDataWords16(typeDmaBlock, block);
```

```
    unsigned int subaddress = mil1553ud_dmaBlock_tdModeSubaddress(block);
    unsigned int activeBus = mil1553ud_dmaBlock_getActiveBus(typeDmaBlock, block);
    emit signal_writeInTerminal(
                QString("Транзакция - тип = %1 - %2 - freeTimer: %3 -
CntServWords: %4 - CntDataWords: %5 - Subaddress: %6 - Шина: %7")
                .arg(QString::number(typeTransaction))
                .arg( (isSuccessTransaction) ? "нет ошибок" : "ошибка" )
                .arg(QString::number(freeTimer))
                .arg(QString::number(countServiceWords16))
                .arg(QString::number(countDataWords16))
                .arg(QString::number(subaddress))
                .arg( (activeBus == MIL1553_BUS_A) ? "Шина А" : "Шина Б" )
                                );
    if (countDataWords16 > 0) {
        unsigned short* dataWord16 = (unsigned
short*)mil1553ud_dmaBlock_getPtrDataPart(block);
        QString dataWordsString = QString("Слова данных:");
        for(unsigned int i = 0; i < countDataWords16; i++) {
            if (i == countDataWords16-1) {
                dataWordsString += QString(" 0x%1;")
                        .arg(QString::number(dataWord16[i], 16));
            } else {
                dataWordsString += QString(" 0x%1,")
                        .arg(QString::number(dataWord16[i], 16));
            }
        }
        emit signal_writeInTerminal(dataWordsString);
    }
}
```

ПРИЛОЖЕНИЕ В (ИНФОРМАЦИОННОЕ)

Скриншоты работы тестовой программы

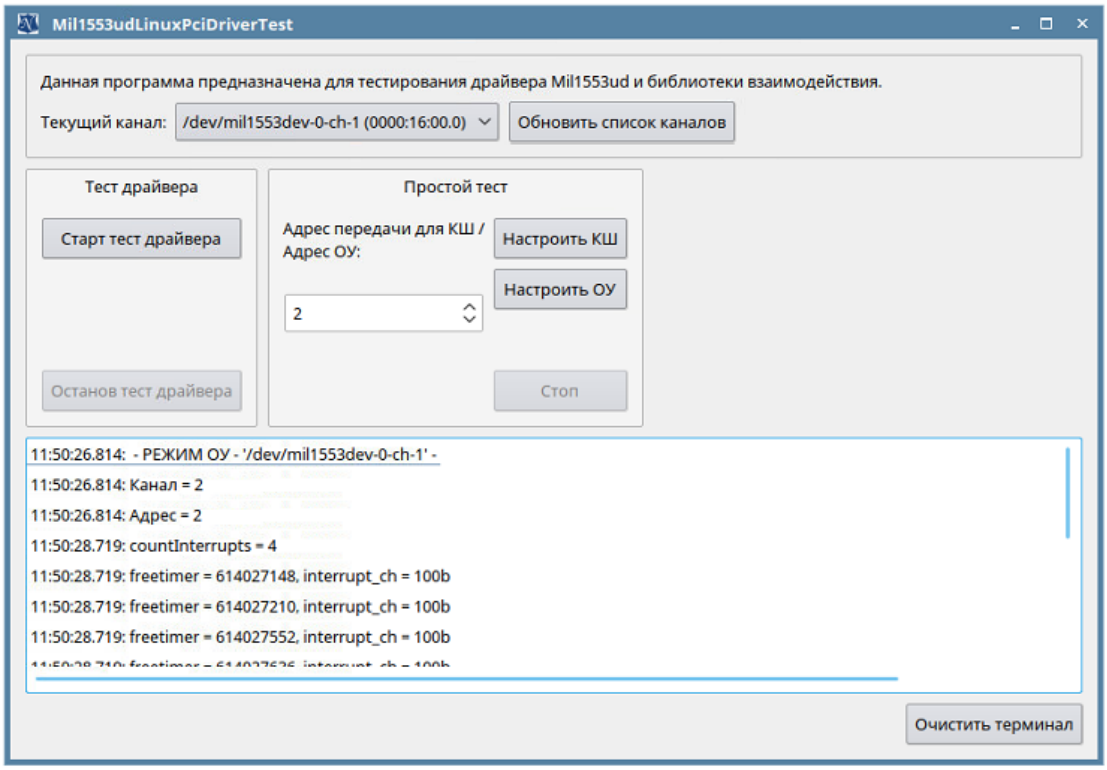


Рисунок Б1 – Канал в режиме КШ

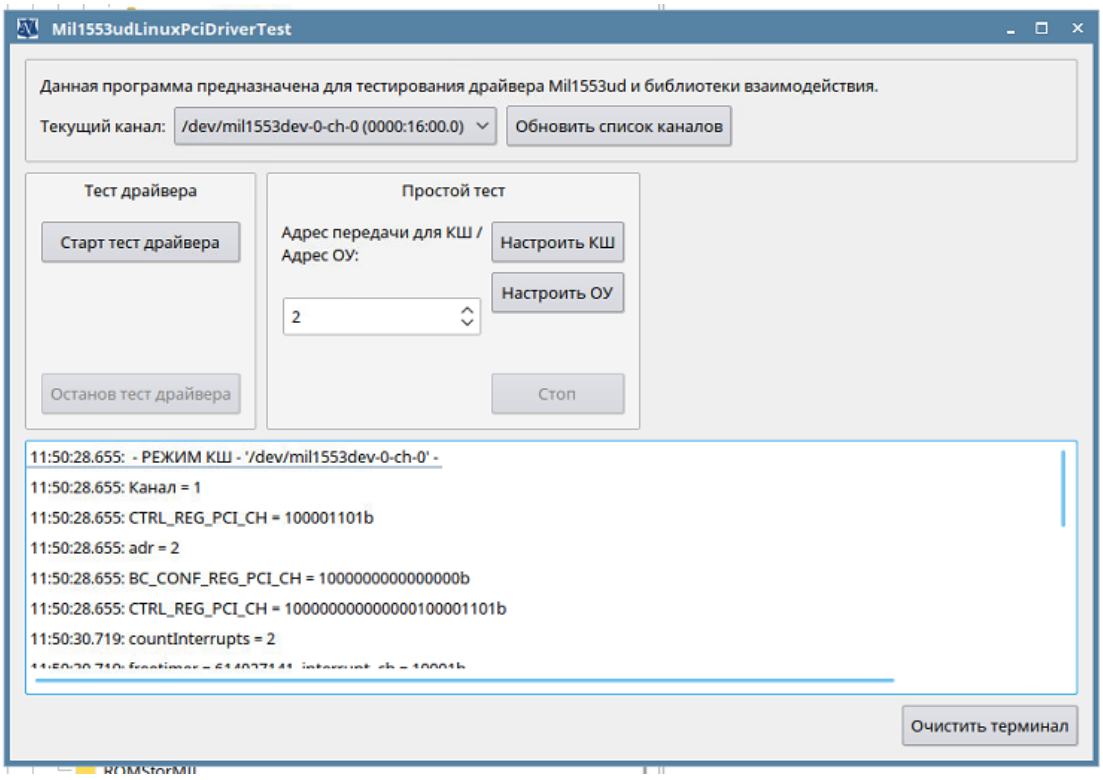


Рисунок Б2 – Канал в режиме ОУ

# 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 | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |