УТВЕРЖДАЮ

Генеральный директор

ООО «НОВОМАР»

_____ Т.В. Буга

«____»_____2021 г.

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

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

(ОС LINUX)

Руководство программиста

ЛИСТ УТВЕРЖДЕНИЯ

RU.МСКЮ.20101-05 33 01-ЛУ

От	Инженер-программист
_____	_____ _____
«____»_____2021 г.	«____»_____2021 г.

2021

Из	*Под*	*Дат*

Литера

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

RU.МСКЮ.20101-05 33 01

Утвержден

RU.МСКЮ.20101-05 33 01-ЛУ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«ДРАЙВЕР MIL1553UD» И «БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ MIL1553UD»

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

(ОС LINUX)

Руководство программиста

RU.МСКЮ.20101-05 33 01

Листов 75

2021

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 блоков 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.

Примечание: вновь добавленные функции имеют префикс mil1553_ (с пункта 4.2.28), первоначальные функции имеют префикс mil1553ud_.

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 Считывает блоки прерываний, готовые к чтению, из канала

Описание функции приведено на рисунке 36.

```
///
/// \brief mil1553ud_readInterruptBlocks считывает блоки прерываний,
готовые к чтению, из канала
/// \param fd дескриптор файла
/// \param block указатель на контейнер
/// \return результат операции
///
unsigned int mil1553ud_readInterruptBlocks(int fd,
INTERRUPT_BLOCK_BUFFER* block);
```

Рисунок 36 – Листинг функции

4.2.9 Управление микропрограммой в режиме КШ

Описание функции приведено на рисунке 37.

```
///
/// \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 37 – Листинг функции

4.2.10 Установить стартовый адрес инструкции микропрограммы КШ

Описание функции приведено на рисунке 38.

```
///
/// \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 38 – Листинг функции

4.2.11 Создать контейнер для микропрограммы КШ

Описание функции приведено на рисунке 39.

```
///
/// \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 39 – Листинг функции

4.2.12 Освободить контейнер для микропрограммы КШ

Описание функции приведено на рисунке 40.

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

Рисунок 40 – Листинг функции

4.2.13 Формирование инструкции для микропрограммы КШ

Описание функции приведено на рисунке 41.

```
///
/// \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 41 – Листинг функции

4.2.14 Формирование нечётной части операции для микропрограммы КШ

Описание функции приведено на рисунке 42.

```
///
/// \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 42 – Листинг функции

4.2.15 Формирование чётной части операции для микропрограммы КШ

Описание функции приведено на рисунке 43.

```
///
/// \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 43 – Листинг функции

4.2.16 Формирование командного слова операции для микропрограммы КШ

Описание функции приведено на рисунке 44.

```
///
/// \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 44 – Листинг функции

4.2.17 Получить указатель на блок DMA из контейнера

Описание функции приведено на рисунке 45.

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

Рисунок 45 – Листинг функции

4.2.18 Проверка на допустимость контейнера DMA

Описание функции приведено на рисунке 46.

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

Рисунок 46 – Листинг функции

4.2.19 Получить тип транзакции из блока DMA

Описание функции приведено на рисунке 47.

```
///
/// \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 47 – Листинг функции

4.2.20 Проверить успешность транзакции

Описание функции приведено на рисунке 48.

```
///
/// \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 48 – Листинг функции

4.2.21 Получить количество служебных 16-разрядных слов

Описание функции приведено на рисунке 49.

```
///
/// \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 49 – Листинг функции

4.2.22 Получить количество 16-разрядных слов данных

Описание функции приведено на рисунке 50.

```
///
/// \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 50 – Листинг функции

4.2.23 Получить активную шину из блока DMA

Описание функции приведено на рисунке 51.

```
///
/// \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 51 – Листинг функции

4.2.24 Получить указатель на часть с данными блока DMA

Описание функции приведено на рисунке 52.

```
///
/// \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 52 – Листинг функции

4.2.25 Получить подадрес из блока DMA в режиме ОУ

Описание функции приведено на рисунке 53.

```
///
/// \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 53 – Листинг функции

4.2.26 Открыть канал (символьное устройство) на чтение и запись

Описание функции приведено на рисунке 54.

```
/// \brief mil1553_open открыть канал
/// \param _file имя файла символьного устройства канала
/// \return дескриптор открытого канала
#define mil1553_open(_file) \
    ::open(_file, O_RDWR)
```

Рисунок 54 – Листинг функции

4.2.27 Открыть канал (символьное устройство)

Описание функции приведено на рисунке 55.

```
/// \brief mil1553_openfl открыть канал
/// \param _file имя файла символьного устройства канала
/// \param _oflags флаги-опции открытия
/// \return дескриптор открытого канала
#define mil1553_openfl(_file, _oflags) \
    ::open(_file, _oflags)
```

Рисунок 55 – Листинг функции

4.2.28 Закрыть канал (символьное устройство)

Описание функции приведено на рисунке 56.

```
/// \brief mil1553_close закрыть канал
/// \param _fd дескриптор открытого канала
#define mil1553_close(_fd) \
    ::close(_fd)
```

Рисунок 56 – Листинг функции

4.2.29 Чтение блоков из ДМА буфера канала

Описание функции приведено на рисунке 57.

```
/// \brief размер блока дма mil1553
#define MIL1553_DMA_BLOCK_SIZE      128
///
/// \brief The TMIL1553_DMA_BLOCK struct блок дма mil1553
///
struct TMIL1553_DMA_BLOCK {
    unsigned char mData[MIL1553_DMA_BLOCK_SIZE];        ///<
содержимое блока
};

/// \brief максимальный размер контейнера блоков дма mil1553
#define MIL1553_DMA_CONTAINER_BLOCKS_MAX_SIZE   128
///
/// \brief The MIL1553_DMA_CONTAINER_BLOCKS struct контейнер блоков
дма mil1553
///
struct MIL1553_DMA_CONTAINER_BLOCKS {
    struct TMIL1553_DMA_BLOCK
mBlocks[MIL1553_DMA_CONTAINER_BLOCKS_MAX_SIZE];   ///< блоки ДМА
    int mCount; ///< кол-во блоков
};
///
/// \brief mil1553_readDma Чтение данных (блоков) из ДМА буфера
канала (макс - 128 блоков за раз)
/// \param fd дескриптор канала
/// \param container контейнер блоков ДМА
/// \return результат операции
///
unsigned int mil1553_readDma(int fd, struct
MIL1553_DMA_CONTAINER_BLOCKS* container);
```

Рисунок 57 – Листинг функции

4.2.30 Включение/выключение канала

Описание функции приведено на рисунке 58.

```
///
/// \brief mil1553_manChannel Вкл./выкл. канала
/// \param fd дескриптор канала
/// \param op MIL_DEV_CHANNEL_ON, MIL_DEV_CHANNEL_OFF
/// \return результат операции
///
unsigned int mil1553_manChannel(int fd, unsigned int op);
```

Рисунок 58 – Листинг функции

4.2.31 Выбор роли (режима) канала

Описание функции приведено на рисунке 59.

```
///
/// \brief mil1553_setRole Выбор роли (режима) канала КШ, ОУ, МШ,
ОУМШ
/// \param fd дескриптор канала
/// \param role роль (режим)
/// \return результат операции
///
unsigned int mil1553_setRole(int fd, unsigned int role);
```

Рисунок 59 – Листинг функции

4.2.32 Получить количество каналов со всех девайсов

Описание функции приведено на рисунке 60.

```
///
/// \brief getCountChannels получить кол-во каналов со всех девайсов
/// \return кол-во каналов
///
int mil1553_getCountChannels(void);
```

Рисунок 60 – Листинг функции

4.2.33 Получить количество девайсов

Описание функции приведено на рисунке 61.

```
///
/// \brief mil1553_getCountDevices получить количество девайсов
/// \return кол-во девайсов
///
int mil1553_getCountDevices(void);
```

Рисунок 61 – Листинг функции

4.2.34 Получить массив описателей девайсов

Описание функции приведено на рисунке 62.

```
///
/// \brief mil1553_getArrayDevices получить массив описателей
девайсов
/// выделяет память под массив
/// \param devices указатель на массив описателей девайсов
/// \param length указатель на длину массива
///
void mil1553_getArrayDevices(struct TMil1553Device** devices, int*
length);
```

Рисунок 62 – Листинг функции

4.2.35Получить номер девайса, к которому принадлежит канал

Описание функции приведено на рисунке 63.

```
///
/// \brief mil1553_getNumberDevice получить номер устройства, к
которому принадлежит канал
/// \param channel канал
/// \return номер устройства
///
int mil1553_getNumberDevice(struct TMil1553Channel * channel);
```

Рисунок 63 – Листинг функции

4.2.36Получить список каналов всех устройств

Описание функции приведено на рисунке 64.

```
///
/// \brief mil1553_getArrayChannels получить список каналов всех
устройств
/// \param channels каналы
/// \param length размер массива
///
void mil1553_getArrayChannels(struct TMil1553Channel **channels, int
*length);
```

Рисунок 64 – Листинг функции

4.2.37Получить список каналов заданного девайса

Описание функции приведено на рисунке 65.

```
///
/// \brief mil1553_getArrayChannelsByDevice Получить список каналов
у заданного девайса
/// \param dev информация о девайсе
/// \param channels каналы
/// \param length размер массива
///
void mil1553_getArrayChannelsByDevice(struct TMil1553Device* dev,
struct TMil1553Channel **channels, int *length);
```

Рисунок 65 – Листинг функции

4.2.38Записать значение в регистр

Описание функции приведено на рисунке 66.

```
///
/// \brief mil1553_writeReg записать значение регистра
/// \param fd дескриптор канала
/// \param reg инфо о регистре
/// \return результат операции
///
unsigned int mil1553_writeReg(int fd, SADDR_DATA* reg);
```

Рисунок 66 – Листинг функции

4.2.39Прочитать значение из регистра

Описание функции приведено на рисунке 67.

```
///
/// \brief mil1553_readReg прочитать значение регистра
/// \param fd дескриптор канала
/// \param reg инфо о регистре
/// \return результат операции
///
unsigned int mil1553_readReg(int fd, SADDR_DATA* reg);
```

Рисунок 67 – Листинг функции

4.2.40Изменить заданные биты регистра

Описание функции приведено на рисунке 68.

```
///
/// \brief mil1553_modifyReg изменить заданные биты регистра
/// \param fd дескриптор канала
/// \param reg инфо о регистре
/// \return результат операции
///
unsigned int mil1553_modifyReg(int fd, SADDR_DATA_BIT_MASK* reg);
```

Рисунок 68 – Листинг функции

4.2.41Получить информацию о девайсе

Описание функции приведено на рисунке 69.

```
///
/// \brief mil1553_getDeviceInfo получить информацию о плате
/// \param fd дескриптор канала
/// \param info инфо о канале и плате
/// \return результат операции
///
unsigned int mil1553_getDeviceInfo(int fd, VERSION* info);
```

Рисунок 69 – Листинг функции

4.2.42Получить версию драйвера

Описание функции приведено на рисунке 70.

```
///
/// \brief mil1553_getDriverVersion получить версию драйверу
/// \param fd дескриптор канала
/// \param version версия драйвера
/// \return результат операции
///
unsigned int mil1553_getDriverVersion(int fd, unsigned int*
version);
```

Рисунок 70 – Листинг функции

4.2.43Получить информацию о PCI локации девайса

Описание функции приведено на рисунке 71.

```
///
/// \brief mil1553_getDevicePciInfo получить информацию о pci
локации платы
/// \param fd дескриптор канала
/// \param pciInfo информация о pci локации платы
/// \return результат операции
///
unsigned int mil1553_getDevicePciInfo(int fd, PCI_LOCATION*
pciInfo);
```

Рисунок 71 – Листинг функции

4.2.44 Сброс ДМА

Описание функции приведено на рисунке 72.

```
///
/// \brief mil1553_clearDma Сброс ДМА
/// \param fd дескриптор канала
/// \return результат операции
///
unsigned int mil1553_clearDma(int fd);
```

Рисунок 72 – Листинг функции

4.2.45 Управление ДМА

Описание функции приведено на рисунке 73.

```
///
/// \brief mil1553_manDma управление ДМА
/// \param fd дескриптор канала
/// \param op 0 - выкл, !0 - вкл
/// \return результат операции
///
unsigned int mil1553_manDma(int fd, int op);
```

Рисунок 73 – Листинг функции

4.2.46 Выбор шины А и/или Б

Описание функции приведено на рисунке 74.

```
///
/// \brief mil1553_switchBus Выбор шины А и/или Б
/// \param fd дескриптор канала
/// \param bus шина - MIL_BUS_A_EN, MIL_BUS_B_EN
/// \return результат операции
///
unsigned int mil1553_switchBus(int fd, unsigned int bus);
```

Рисунок 74 – Листинг функции

4.2.47Запись данных в RAM КШ

Описание функции приведено на рисунке 75.

```
///
/// \brief The EMil1553BcTypeDataRam enum тип озу контроллера шины
///
enum EMil1553BcTypeDataRam {
    Mil1553BcTypeInstructions = 0,    ///< инструкции кш
    Mil1553BcTypeOperations,          ///< операции кш
    Mil1553BcTypeData                 ///< данные кш
};

/// \brief максимальный размер массива слов озу кш не данные
#define MIL1553_BC_MAX_SIZE_RAM_NONDATA     4096
/// \brief максимальный размер массива слов озу кш данные
#define MIL1553_BC_MAX_SIZE_RAM_DATA        (4096*2)

/// \brief ошибка записи в BC RAM - ошибка максимальной длины данных
#define MIL1553_BC_WRT_DATA_RAM_ERROR_MAX_LEN       -2
/// \brief ошибка записи в BC RAM - ошибка записи одного блока RAM
#define MIL1553_BC_WRT_DATA_RAM_ERROR_BAD_WRT       -3
/// \brief ошибка записи в BC RAM - ошибка чтения одного блока RAM
#define MIL1553_BC_WRT_DATA_RAM_ERROR_BAD_RD        -4
/// \brief ошибка записи в BC RAM - ошибка контроля записи одного
блока RAM
#define MIL1553_BC_WRT_DATA_RAM_ERROR_BAD_CHECK     -5

///
/// \brief mil1553_bc_writeDataToRam запись данных в RAM контроллера
шины
/// \param fd дескриптор канала
/// \param type тип озу контроллера шины
/// \param data данные (массив)
/// \param length размер массива
/// \return результат операции
///
unsigned int mil1553_bc_writeDataToRam(int fd, enum
EMil1553BcTypeDataRam type, unsigned int* data, int length);
```

Рисунок 75 – Листинг функции

4.2.48 Разрешение/запрет подадресов ОУ на приём/передачу

Описание функции приведено на рисунке 76.

```
///
/// \brief mil1553_rt_manSubaddresses Разрешение подадресов ОУ на
приём/передачу
/// \param fd дескриптор канала
/// \param direction направление MIL_RT_BUF_RECEIVE,
MIL_RT_BUF_TRANSMIT
/// \param subadr_mask маска подадресов (1 - изменять подадрес, 0 -
не изменять поадрес) (с 1 по 30 подадрес)
/// \param action_mask маска действий (1 - разрешить, 0 - запретить)
(с 1 по 30 подадрес)
/// \return результат операции
///
unsigned int mil1553_rt_manSubaddresses(int fd, unsigned int
direction, unsigned int subadr_mask, unsigned int action_mask);
```

Рисунок 76 – Листинг функции

4.2.49 Установить адрес ОУ

Описание функции приведено на рисунке 77.

```
///
/// \brief mil1553_rt_setAddress установить адрес ОУ
/// \param fd дескриптор канала
/// \param address адрес ОУ
/// \return результат операции
///
unsigned int mil1553_rt_setAddress(int fd, unsigned char address);
```

Рисунок 77 – Листинг функции

4.2.50 Управление режимом буфера ОУ на передачу

Описание функции приведено на рисунке 78.

```
///
/// \brief mil1553_rt_manSubadressBufferMode управление режимом
буфера ОУ на передачу
/// \param fd дескриптор канала
/// \param subaddress подадрес
/// \param mode режим буфера MIL_RT_MODE_PROG, MIL_RT_MODE_HRDW
/// \return результат операции
///
unsigned int mil1553_rt_manSubadressTransmitBufferMode(int fd,
unsigned char subaddress, unsigned int mode);
```

Рисунок 78 – Листинг функции

4.2.51 Управление готовностью буфера ОУ на передачу

Описание функции приведено на рисунке 79.

```
///
/// \brief mil1553_rt_manSubaddressTransmitBufferReady управление
готовностью буфера ОУ на передачу
/// \param fd дескриптор канала
/// \param subaddress подадрес
/// \param cmd_buf команды управления буферами
/// \return результат операции
///
unsigned int mil1553_rt_manSubaddressTransmitBufferReady(int fd,
unsigned char subaddress, unsigned int cmd_buf);
```

Рисунок 79 – Листинг функции

4.2.52 Запись данных в подадрес отправного буфера ОУ

Описание функции приведено на рисунке 80.

```
/// \brief кол-ва слов в подадресе
#define MIL1553_RT_SUBADDRESS_COUNT_WORDS   32

///
/// \brief The TRtSubaddressData struct данные в подадресе
///
struct TRtSubaddressData {
    unsigned short mWords[MIL1553_RT_SUBADDRESS_COUNT_WORDS];   ///<
16-битные слова
    int mCount;                                                 ///<
кол-во слов
};

///
/// \brief mil1553_rt_writeDataToSubaddressTransmit запись данных в
подадрес отправного буфера ОУ
/// \param fd дескриптор канала
/// \param num_buf номер аппаратного буфера MIL_RT_BUF0, MIL_RT_BUF1
/// \param subaddress подадрес
/// \param data данные
/// \return результат операции
///
unsigned int mil1553_rt_writeDataToSubaddressTransmit(int fd,
unsigned int num_buf, unsigned char subaddress, struct
TRtSubaddressData data);
```

Рисунок 80 – 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 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

5.2 БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ MIL1553UD

Описание кодов ошибок приведено на рисунке 84.

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

Рисунок 84 - 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

Листинг Б1 – Листинг программы

```
#ifndef CTEST1_H
#define CTEST1_H

#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <time.h>
#include <unistd.h>
#include <strings.h>
#include <errno.h>
#include <ctype.h>
#include <time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/time.h>

#include "global.h"

/// \brief кол-во каналов, учавствующих в тесте
#define COUNT_CH (8+1)
/// \brief шахматка
#define CHESS_DATA          0xAAAA
/// \brief обратная шахматка
#define REVERSE_CHESS_DATA  0x5555
/// \brief индекс канала КШ в массиве параметров
#define BC_CH_INDEX         0

///
/// \brief The SConfigCh struct каналы, готовые к работе
/// 0 - КШ, 1 - ОУ1 ...
///
struct SConfigCh {
    int mIsReady[COUNT_CH];     ///< признак готовности канала
};
///
/// \brief The TStatChCTest1 struct статистика канала
///
struct TStatChCTest1 {
    int mDmaBlk;    ///< кол-во дма блоков
    int mErr;       ///< кол-во дма блоков, с ошибкой
};
///
/// \brief The TStatCTest1 struct статистика теста
///
struct TStatCTest1 {
    struct TStatChCTest1 mCh[COUNT_CH];     ///< поканальная статистика
};

///
/// \brief The TParamCTest1 struct параметры теста 1
///
```

```
struct TParamCTest1 {
    int mDurationHours;     ///< продолжительность теста - часы
    int mDurationMinutes;   ///< продолжительность теста - минуты
    int mLine;              ///< линия: 0-A; 1-B
    char *mNames[COUNT_CH][CHANNEL_NAME_LENGTH];
};
/// \brief неиспользуемй канал
#define NULL_DEV    "NULL"
#define FALSE_VAL   0
#define TRUE_VAL    (~(FALSE_VAL))

///< продолжительность теста - часы
#define T_CTEST1_DURATION_H     "duration_h"
///< продолжительность теста - минуты
#define T_CTEST1_DURATION_M     "duration_m"
///< линия
#define T_CTEST1_LINE           "line"
///< КШ
#define T_CTEST1_BC             "bc"
///< ОУ1
#define T_CTEST1_RT1            "rt1"
///< ОУ2
#define T_CTEST1_RT2            "rt2"
///< ОУ3
#define T_CTEST1_RT3            "rt3"
///< ОУ4
#define T_CTEST1_RT4            "rt4"
///< ОУ5
#define T_CTEST1_RT5            "rt5"
///< ОУ6
#define T_CTEST1_RT6            "rt6"
///< ОУ7
#define T_CTEST1_RT7            "rt7"
///< ОУ8
#define T_CTEST1_RT8            "rt8"


extern char configFileName[];                   ///< имя файла конфигурации,
считанного из аргументов
extern char fullConfigFileName[];               ///< полное имя файла конфигурации
в папке config
extern struct TParamCTest1 params;              ///< параметры теста 1
extern struct TMil1553Channel* channels;        ///< каналы
extern struct TMil1553Device* devices;          ///< девайсы
extern struct SConfigCh configChannels;         ///< конфигкрация готовности
каналов
extern struct TStatCTest1 statTest;             ///< статистика теста

///
/// \brief main точка входа в программу
/// \param argc кол-во аргументов
/// \param argv аргументы
/// \return код возврата
///
int main(int argc, char* argv[]);
///
/// \brief constructFullConfigFileName построить полное имя файла конфигурации
/// \param fullname полное имя
/// \param size размер
/// \param name имя
///
```

```
void constructFullConfigFileName(char* fullname, int size, char* name);
///
/// \brief readConfigFile чтение конфигурации из файла
/// \param fullname полное имя файла конфигурации
/// \param params параметры теста
/// \return код результата
///
int readConfigFile(char* fullname, struct TParamCTest1* params);
///
/// \brief procCTest1 процедура консольного теста 1
/// \param params параметры теста
///
void procCTest1(struct TParamCTest1 params);
///
/// \brief initLogCTest1 настройка логов теста 1
///
void initLogCTest1(void);
///
/// \brief initDefaultParamsCTest1 инициализация параметров ао-умолчанию теста 1
/// \param params параметры теста
///
void initDefaultParamsCTest1(struct TParamCTest1* params);

#endif // CTEST1_H


#include "ctest1.h"
#include <stdio.h>
#include <stdlib.h>

char configFileName[MAX_LEN_CONFIG_FILE];
char fullConfigFileName[MAX_LEN_FULL_CONFIG_FILE];
struct TParamCTest1 params;
struct TMil1553Channel* channels;
int channelsCount;
struct TMil1553Device* devices;
int devicesCount;
struct SConfigCh configChannels; ///< конфигкрация готовности каналов
struct TStatCTest1 statTest; ///< статистика теста

static unsigned int isBcStopped;

#define PRINT_LINE() \
    LOGGER_PRINT("%s\n", "--- --- ---")
/// \brief стартовый адрес инструкции КШ
#define BC_START_ADDRESS 0
/// \brief групповая команда синхронизации
#define TR_SYNCRO 9

int main(int argc, char* argv[])
{
    initLogCTest1();
    LOGGER_PRINT("%s %s\n","ctest1 - version", VERSION_APP);
    int ret = GOOD_CODE;
    if (argc != 2) { // проверка кол-ва аргументов
        usage:
        LOGGER_PRINT("%s\n","Usage: ctest1 <configfilename>");
        ret = BAD_CODE;
    } else { // кол-во аргументов - ок
        if (sscanf(argv[1], "%s", configFileName) != 1) { // чтение имени
конфигурационного файла
```

```
            LOGGER_PRINT("%s\n", "invalid configfilename\n");
            goto usage;
        } else { //имя конфигурационного файла - ок
            constructFullConfigFileName(fullConfigFileName,
MAX_LEN_FULL_CONFIG_FILE, configFileName);
            initDefaultParamsCTest1(&params);
            ret = readConfigFile(fullConfigFileName, &params); // чтение параметров
из конфигурационного файла
            if (ret == BAD_CODE) { //параметры из конфигурационного файла - error
                LOGGER_PRINT("Can not read test params from '%s' \n",
fullConfigFileName);
            } else { //параметры из конфигурационного файла - ок
                procCTest1(params);
            }
        }
    }
    LOGGER_FLUSH();
    LOGGER_CLOSE();
    return ret;
}
/// \brief размер мем файла
#define MEM_SIZE    1024
/// \brief инструкции КШ
static unsigned int memDesc[MEM_SIZE];
/// \brief операции КШ
static unsigned int memOp[MEM_SIZE];
/// \brief данные КШ
static unsigned int memData[MEM_SIZE];

void loadMemFiles(void) {
    loadMemTxtFile("./mem/consol_dat_ram.mem", memData, MEM_SIZE);
    loadMemTxtFile("./mem/consol_inst_ram.mem", memDesc, MEM_SIZE);
    loadMemTxtFile("./mem/consol_op_ram.mem", memOp, MEM_SIZE);
}
/// \brief кол-во подциклов в мем файле инструкций
#define MAX_DESC_SHIFT_COUNTER  6
/// \brief стартовые адреса подциклов
unsigned int cycleAddresses[MAX_DESC_SHIFT_COUNTER] = { 0x10, 0x20, 0x30, 0x40,
0x50, 0x60 };

void memDescRtOff(unsigned int * word) {
    int countOfOnes = 0;
    *word = (1<<20) | *word;
    for (int i=16; i<=30; i++) {
        if (*word & (1<<i)) { countOfOnes++; }
    }
    if (( countOfOnes % 2 ) == 0)
        *word = *word | (1u<<31);
    else
        *word = *word & ~(1u<<31);
}

void memDescRtOffWithCheckReady(int num_rt, int counter) {
    if (FALSE_VAL == configChannels.mIsReady[num_rt]) {
        unsigned int adr = cycleAddresses[counter] + num_rt - 1;
        memDescRtOff(&(memDesc[adr]));
    }
}

void applyParametresToMem(void) {
    int counter;
```

```
    if (0 == params.mLine) { // off line b
        memDescRtOff(&memDesc[0x6]);
        memDescRtOff(&memDesc[0x7]);
        memDescRtOff(&memDesc[0x8]);
        memDescRtOff(&memDesc[0x9]);
    } else { // off line a
        memDescRtOff(&memDesc[0x2]);
        memDescRtOff(&memDesc[0x3]);
        memDescRtOff(&memDesc[0x4]);
        memDescRtOff(&memDesc[0x5]);
    }
    for (counter = 0; counter < MAX_DESC_SHIFT_COUNTER; counter++) {
        // применение параметров к мем descriptors
        memDescRtOffWithCheckReady(1, counter);
        memDescRtOffWithCheckReady(2, counter);
        memDescRtOffWithCheckReady(3, counter);
        memDescRtOffWithCheckReady(4, counter);
        memDescRtOffWithCheckReady(5, counter);
        memDescRtOffWithCheckReady(6, counter);
        memDescRtOffWithCheckReady(7, counter);
        memDescRtOffWithCheckReady(8, counter);
    }
}

void writeMemToBc(void) {
    if (configChannels.mIsReady[0]) {
        int isGood;
        int fd = mil1553_open(params.mNames[0]);
        isGood = mil1553_bc_writeDataToRam(fd, Mil1553BcTypeInstructions, memDesc,
MEM_SIZE);
        LOGGER_PRINT_STR("./mem/consol_inst_ram.mem");
        if (isGood == GOOD_CODE) {
           LOGGER_PRINT_STR(" was writed.\n");
        } else {
           LOGGER_PRINT_STR(" was NOT writed.\n");
        }
        isGood = mil1553_bc_writeDataToRam(fd, Mil1553BcTypeOperations, memOp,
MEM_SIZE);
        LOGGER_PRINT_STR("./mem/consol_op_ram.mem");
        if (isGood == GOOD_CODE) {
           LOGGER_PRINT_STR(" was writed.\n");
        } else {
           LOGGER_PRINT_STR(" was NOT writed.\n");
        }
        isGood = mil1553_bc_writeDataToRam(fd, Mil1553BcTypeData, memData,
MEM_SIZE);
        LOGGER_PRINT_STR("./mem/consol_dat_ram.mem");
        if (isGood == GOOD_CODE) {
           LOGGER_PRINT_STR(" was writed.\n");
        } else {
           LOGGER_PRINT_STR(" was NOT writed.\n");
        }
        mil1553_close(fd);
    }
}

void initChannel_switchOffChannel(int fd) {
    unsigned int opCh = MIL_DEV_CHANNEL_OFF;
    mil1553_manChannel(fd, &opCh);
    unsigned int opDma = MIL1553_OFF;
    mil1553_manDma(fd, opDma);
```

```
    mil1553_clearDma(fd);
}

void initChannel_config(int fd, int index) {
    unsigned int modeBus;   // режим
    unsigned int bus = MIL_BUS_A_EN | MIL_BUS_B_EN;   // шина
    if (0 == index) {
        modeBus = MIL_MODE_BUS_CONTR; // режим КШ
    } else {
        modeBus = MIL_MODE_TERMINAL_DEV;
    }
    mil1553_setRole(fd, modeBus);
    mil1553_switchBus(fd, bus);
    if (0 != index) {
        mil1553_rt_setAddress(fd, index);
    }
}

void initChannel_initRtSubaddresses(int fd) {
    mil1553_rt_manSubaddresses(fd, MIL_RT_BUF_RECEIVE, 0x7FFFFFFE /* меняем с 1 по
30 подадрес*/, 0x7FFFFFFE /* включаем на приём с 1 по 30 подадрес*/);
    mil1553_rt_manSubaddresses(fd, MIL_RT_BUF_TRANSMIT, 0x7FFFFFFE /* меняем с 1
по 30 подадрес*/, 0x7FFFFFFE /* включаем на приём с 1 по 30 подадрес*/);
}

void initChannel_initRtSubaddressBuffer(int fd) {
    unsigned char subaddress;
    for (subaddress = 1; subaddress <= 30; subaddress++) {
        mil1553_rt_manSubadressTransmitBufferMode(fd, subaddress,
MIL_RT_MODE_PROG);
        mil1553_rt_manSubaddressTransmitBufferReady(fd, subaddress,
MIL_RT_BUF_0_ON_1_OFF);
    }
    generateDataRt(fd);
}

void initChannel_switchOnChannel(int fd) {
    // включаем ДМА
    unsigned int opDma = MIL1553_ON;
    mil1553_manDma(fd, opDma);
    // включаем канал
    unsigned int opCh = MIL_DEV_CHANNEL_ON;
    mil1553_manChannel(fd, opCh);
}

void initChannels(void) {
    applyParametresToMem(); // применение парметров к мем файлу с инструкциями
    int index;
    for (index = 0; index < COUNT_CH; index++) { // цикл по каналам
        if (configChannels.mIsReady[index]) {
            int fd = mil1553_open(params.mNames[index]);
            initChannel_switchOffChannel(fd); // выкл канала
            initChannel_config(fd, index); // конфигурация канала
            if (0 != index) { // для ОУ
                initChannel_initRtSubaddresses(fd); // установка разрешений
подадресов
                initChannel_initRtSubaddressBuffer(fd); // установка буферов на
передачу
            }
            initChannel_switchOnChannel(fd); // вкл канала
            mil1553_close(fd);
```

```
        }
    }
    writeMemToBc(); // запись мем файлов в ОЗУ КШ
}

void deinitChannels(void) {
    int index;
    for (index = 0; index < COUNT_CH; index++) { // цикл по каналам
        if (configChannels.mIsReady[index]) {
            int fd = mil1553_open(params.mNames[index]);
            // деинициализация
            unsigned int opCh = MIL_DEV_CHANNEL_OFF;
            mil1553_manChannel(fd, &opCh);
            unsigned int opDma = MIL1553_OFF;
            mil1553_manDma(fd, opDma);
            mil1553_clearDma(fd);
            mil1553_close(fd);
        }
    }
}

void startBc(void) {
    if (configChannels.mIsReady[0]) { // старт канала КШ, если он есть
        int fd = mil1553_open(params.mNames[0]);
        mil1553ud_bcSetStartInstructionAddress(fd, BC_START_ADDRESS);
        mil1553ud_bcProgram(fd, BC_PROGRAM_START); // BCSTRT - старт выполнения
программы КШ;
        mil1553_close(fd);
    }
}

void stopBc(void) {
    if (configChannels.mIsReady[0]) { // старт канала КШ, если он есть
        int fd = mil1553_open(params.mNames[0]);
        mil1553ud_bcProgram(fd, BC_PROGRAM_STOP); // BCSTRT - старт выполнения
программы КШ;
        mil1553_close(fd);
    }
}

void constructFullConfigFileName(char* fullname, int size, char* name) {
    snprintf(fullname, size, "%s%s", STR_CONFIG, name);
}

#define READ_CHANNEL_NAME(num) \
    sscanf(str, "%s %s", typeArgStr, valStr); \
    if (0 == strncmp (valStr, NULL_DEV, sizeof(NULL_DEV))) { \
        configChannels.mIsReady[num] = FALSE_VAL; \
        memcpy(params->mNames[num], valStr, CHANNEL_NAME_LENGTH); \
    } else { \
        char fullNameCh[CHANNEL_NAME_LENGTH]; \
        configChannels.mIsReady[num] = TRUE_VAL; \
        snprintf(fullNameCh, CHANNEL_NAME_LENGTH, "%s%s", STR_DEV, valStr); \
        memcpy(params->mNames[num], fullNameCh, CHANNEL_NAME_LENGTH); \
    }

int readConfigFile(char* fullname, struct TParamCTest1* params) {
    FILE* file = fopen(fullname, "r");
    char* str;
    if (file == NULL) {
        LOGGER_PRINT("Can not open config file '%s'\n", fullname);
```

```
        return BAD_CODE;
    }
    LOGGER_PRINT("parsing '%s':\n", fullname);
    while (!feof(file)) {
        char valStr[CHANNEL_NAME_LENGTH];
        readline(file, &str);
        LOGGER_PRINT("%s\n", str); // for debug
        char typeArgStr[30];
        int valArg = BAD_CODE;
        sscanf(str, "%s %s", typeArgStr, valStr);
        if (0 == strncmp (typeArgStr, T_CTEST1_DURATION_H,
sizeof(T_CTEST1_DURATION_H))) {
            sscanf(str, "%s %d", typeArgStr, &valArg);
            params->mDurationHours = valArg;
        } else if (0 == strncmp (typeArgStr, T_CTEST1_DURATION_M,
sizeof(T_CTEST1_DURATION_M))) {
            sscanf(str, "%s %d", typeArgStr, &valArg);
            params->mDurationMinutes = valArg;
        } else if (0 == strncmp (typeArgStr, T_CTEST1_LINE,
sizeof(T_CTEST1_LINE))) {
            sscanf(str, "%s %d", typeArgStr, &valArg);
            params->mLine = valArg;
        } else if (0 == strncmp (typeArgStr, T_CTEST1_BC, sizeof(T_CTEST1_BC))) {
            READ_CHANNEL_NAME(0);
        } else if (0 == strncmp (typeArgStr, T_CTEST1_RT1, sizeof(T_CTEST1_RT1)))
{
            READ_CHANNEL_NAME(1);
        }  else if (0 == strncmp (typeArgStr, T_CTEST1_RT2, sizeof(T_CTEST1_RT2)))
{
            READ_CHANNEL_NAME(2);
        } else if (0 == strncmp (typeArgStr, T_CTEST1_RT3, sizeof(T_CTEST1_RT3)))
{
            READ_CHANNEL_NAME(3);
        } else if (0 == strncmp (typeArgStr, T_CTEST1_RT4, sizeof(T_CTEST1_RT4)))
{
            READ_CHANNEL_NAME(4);
        } else if (0 == strncmp (typeArgStr, T_CTEST1_RT5, sizeof(T_CTEST1_RT5)))
{
            READ_CHANNEL_NAME(5);
        } else if (0 == strncmp (typeArgStr, T_CTEST1_RT6, sizeof(T_CTEST1_RT6)))
{
            READ_CHANNEL_NAME(6);
        } else if (0 == strncmp (typeArgStr, T_CTEST1_RT7, sizeof(T_CTEST1_RT7)))
{
            READ_CHANNEL_NAME(7);
        } else if (0 == strncmp (typeArgStr, T_CTEST1_RT8, sizeof(T_CTEST1_RT8)))
{
            READ_CHANNEL_NAME(8);
        }
    }
    fclose(file);
    LOGGER_PRINT("PARAMS: duration_h= %d, duration_m= %d, line= %d, bc= %s,\n rt1=
%s,\n rt2= %s,\n rt3= %s,\n rt4= %s,\n rt5= %s,\n rt6= %s,\n rt7= %s,\n rt8= %s
\n\n", \
                 params->mDurationHours, params->mDurationMinutes, params->mLine,
params->mNames[0], \
                 params->mNames[1], params->mNames[2], params->mNames[3], params-
>mNames[4], params->mNames[5], params->mNames[6], params->mNames[7], params-
>mNames[8]);
    PRINT_LINE();
    return GOOD_CODE;
```

```
}

void initLogCTest1(void) {
    char filename[LOGGER_FILENAME_LENGTH];
    constructLogFilename(filename, LOGGER_FILENAME_LENGTH, "ctest1");
    LOGGER_OPEN(filename);
}

void initDefaultParamsCTest1(struct TParamCTest1* params) {
    params->mDurationHours = 0;     ///< продолжительность теста - часы
    params->mDurationMinutes = 0;   ///< продолжительность теста - минуты
}

void initTestDuration(struct TParamCTest1* params, unsigned int*
allDurationSeconds, unsigned int* curDurationSeconds) {
    *curDurationSeconds = 0;
    *allDurationSeconds = params->mDurationMinutes * 60 + params->mDurationHours *
60 * 60;
}

static unsigned int dataAdrRtLineA[8] = {0x0, 0x10, 0x20, 0x30, 0x40, 0x50, 0x60,
0x70};
static unsigned int dataAdrRtLineB[8] = {0x80, 0x90, 0xA0, 0xB0, 0xC0, 0xD0, 0xE0,
0xF0};

unsigned int isGoodDataRt(struct TMIL1553_DMA_BLOCK* block, int index_ch) {
    if (BC_CH_INDEX != index_ch) { // BC
        // проверка данных ОУ
         unsigned short* dataWord16 = (unsigned
short*)mil1553ud_dmaBlock_getPtrDataPart(block);
         unsigned short* dataWord16Etalon = (params.mLine == 0) ? &(memData[
dataAdrRtLineA[index_ch - 1] ]) : &(memData[ dataAdrRtLineB[index_ch - 1] ]);
         for(unsigned int i = 0; i < 15; i++) {
             if ( dataWord16[i] != dataWord16Etalon[i] )
                 return MIL1553UD_FALSE;
         }
    }
    return MIL1553UD_TRUE;
}

unsigned int isGoodData(struct TMIL1553_DMA_BLOCK* block, int index_ch) {
    if (BC_CH_INDEX == index_ch) { // BC
        // проверка данных КШ
        unsigned short* dataWord16 = (unsigned
short*)mil1553ud_dmaBlock_getPtrDataPart(block);
        for(unsigned int i = 0; i < 7; i++) {
            if (dataWord16[i] != CHESS_DATA)
                return MIL1553UD_FALSE;
        }
    }
    return MIL1553UD_TRUE;
}

int isRun(int fd) {
    SADDR_DATA regBcCondCode;
    unsigned int numChannel = mil1553ud_getNumberChannel(fd);
    regBcCondCode.daddr = MIL_REG(numChannel, GPF_BC_COND_CODE_CH);
    mil1553_readReg(fd, &regBcCondCode);
    return (regBcCondCode.data & (1u<<31)) ? MIL1553UD_TRUE : MIL1553UD_FALSE;
}
```

```
char* stringsTransaction[10] = {
    "формат 1 КШ - ОУ (КС)",
    "формат 2 ОУ - КШ (КС)",
    "формат 3 ОУ - ОУ  (КС)",
    "формат 4 КУ",
    "формат 5 КУ",
    "формат 6 КУ",
    "формат 7 КШ - ОУ групповая  (КС)",
    "формат 8 ОУ - ОУ групповая   (КС)",
    "формат 9 КУ групповая",
    "формат 10 КУ групповая"
};

char* getMilTransactionString(unsigned int typeTransaction){
    return stringsTransaction[typeTransaction-1];
}

char* errorsBcString[16] = {
    "Ошибка таймаута приёма.",
    "Ошибка декодера Манчестера II.",
    "Ошибка синхронизации.",
    "Ошибка последовательности.",
    "Интервал между сообщениями t1 меньше 4 мкс.",
    "Ошибка разрыва данных.",
    "Ошибка чётности адреса ОУ.",
    "КС2 содержит КУ.",
    "Ошибка КУ.",
    "Длина сообщения КС1 /= длине сообщения КС2.",
    "Адрес ОУ1 = ОУ2 (команды ОУ-ОУ).",
    "КУ зарезервирована.",
    "КУ не определена.",
    "Адрес/подадрес запрещён.",
    "Установлен запрет на выполнение принятой КУ.",
    "Принято КС с групповым адресом и битом передача установленным в 1 (кроме
КУ)."
};

void printBcError(char* block) {
    int index;
    unsigned int typeDmaBlock = MIL1553UD_TD_DMA_BLOCK;     // ОУ
    unsigned int countServiceWords16 =
mil1553ud_dmaBlock_getCountServiceWords16(typeDmaBlock, block);
    unsigned short* block16 = (unsigned short*)block;
    int shift16 = 6*2 + countServiceWords16 - 1;
    unsigned int error = block16[shift16];
    LOGGER_PRINT("Error value = 0x%x:\n", error);
    for (index = 0; index < 16; index++) {
        if (error & (1<<index)) {
            LOGGER_PRINT("%s;\n", errorsBcString[index]);
        }
    }
}

void processingBc(void) {
    int index = 0; // инрдекс КШ
    if (configChannels.mIsReady[index]) {
        if (MIL1553UD_FALSE == isBcStopped) {
            int fd = mil1553_open(params.mNames[index]);
            int isRunValue = isRun(fd);
            mil1553_close(fd);
            if (MIL1553UD_FALSE == isRunValue)
```

```
                startBc();
        }
        {
            struct MIL1553_DMA_CONTAINER_BLOCKS container;
            int ret, indexBlk;
            int fd = mil1553_open(params.mNames[index]);
            ret = mil1553_readDma(fd, &container); // чтение dma блоков
            if (GOOD_CODE == ret) {
                for (indexBlk = 0; indexBlk < container.mCount; indexBlk++) {
                    char* block = container.mBlocks[indexBlk].mData;
                    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);
                    unsigned short* cw = (unsigned short*)(block + 5*4);
                    unsigned int isGroupCom = MIL1553UD_FALSE;
                    if ((cw[0] >> 11) == 31) // Групповая команда
                        isGroupCom = MIL1553UD_TRUE;
                    // накопление статистики
                    statTest.mCh[index].mDmaBlk++;
                    if (MIL1553UD_FALSE == isSuccessTransaction)
                    { // проверка на наличие ошибок
                        statTest.mCh[index].mErr++;
                        PRINT_LINE();
                        // печать инфо об ошибке
                        LOGGER_PRINT("%s on BC: ", "Error");
                        LOGGER_PRINT("Freetimer= %u, Type= %d (%s), Bus= %s,
ServiceWords= %d, DataWords= %d\n", freeTimer, typeTransaction,
getMilTransactionString(typeTransaction), (activeBus == MIL1553_BUS_A) ? "A" :
"Б", countServiceWords16, countDataWords16);
                        printBcError(block);
                        if ((typeTransaction == 1) | (typeTransaction == 2)) {
                            if ((countDataWords16 > 0) && (isSuccessTransaction))
{
                                unsigned short* dataWord16 = (unsigned
short*)mil1553ud_dmaBlock_getPtrDataPart(block);
                                LOGGER_PRINT("%s: ", "Data words");
                                for(unsigned int i = 0; i < countDataWords16; i++)
{
                                    if (i == countDataWords16-1) {
                                        LOGGER_PRINT("0x%x; \n", dataWord16[i]);
                                    } else {
                                        LOGGER_PRINT("0x%x,", dataWord16[i]);
                                    }
                                }
                            }
                        }
                    }
                    if (TR_SYNCRO == typeTransaction) {
//                      LOGGER_PRINT("SYNCHRO - RT%d\n", index);
                        generateDataRt(fd);
                    }
```

```
                }
            } else {
                LOGGER_PRINT("ReadDma error in BC (%s) = %d\n",
params.mNames[index], ret);
            }
            mil1553_close(fd);
        }
    }
}

void generateDataRt(int fd) {
    struct TRtSubaddressData data0;
    for (int index = 0; index < 32; index++) {
        data0.mWords[index] = rand();
    }
    data0.mCount = 32;
    unsigned char subaddress;
    for (subaddress = 1; subaddress <= 30; subaddress++) {
        mil1553_rt_writeDataToSubaddressTransmit(fd, MIL_RT_BUF0, subaddress,
data0);
        mil1553_rt_writeDataToSubaddressTransmit(fd, MIL_RT_BUF1, subaddress,
data0);
    }
}

char* errorsRtString[16] = {
    "Ошибка приёма данных.",
    "Ошибка декодера Манчестера II.",
    "Ошибка синхронизации.",
    "Ошибка последовательности.",
    "Интервал между сообщениями t1 меньше 4 мкс.",
    "Ошибка разрыва данных.",
    "Ошибка чётности Манчестер II.",
    "Ошибка таймаута t1.",
    "Выполнено два повтора сообщения.",
    "Выполнен один повтор сообщения.",
    "Нет ответа.",
    "Длина данных, принятых от ОУ не соответствует полю команды.",
    "Ошибка адреса ОУ.",
    "Ошибка формата.",
    "Ошибка длины.",
    "Ошибка самоконтроля данных."
};

void printRtError(char* block) {
    int index;
    unsigned int typeDmaBlock = MIL1553UD_TD_DMA_BLOCK;     // ОУ
    unsigned int countServiceWords16 =
mil1553ud_dmaBlock_getCountServiceWords16(typeDmaBlock, block);
    unsigned short* block16 = (unsigned short*)block;
    int shift16 = 6*2 + countServiceWords16 - 1;
    unsigned int error = block16[shift16];
    LOGGER_PRINT("Error value = 0x%x:\n", error);
    for (index = 0; index < 16; index++) {
        if (error & (1<<index)) {
            LOGGER_PRINT("%s;\n", errorsRtString[index]);
        }
    }
}

void processingRts(void) {
```

```
    int index;
    for (index = 1; index < COUNT_CH; index++) { // цикл по ОУ
        if (configChannels.mIsReady[index]) {
            struct MIL1553_DMA_CONTAINER_BLOCKS container;
            int ret, indexBlk;
            int fd = mil1553_open(params.mNames[index]);
            ret = mil1553_readDma(fd, &container); // чтение dma блоков
            if (GOOD_CODE == ret) {
                for (indexBlk = 0; indexBlk < container.mCount; indexBlk++) {
                    char* block = container.mBlocks[indexBlk].mData;
                    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);
                    // накопление статистики
                    statTest.mCh[index].mDmaBlk++;
                    if (MIL1553UD_FALSE == isSuccessTransaction)
                    { // проверка на наличие ошибок
                        statTest.mCh[index].mErr++;
                        // печать инфо об ошибке
                        PRINT_LINE();
                        LOGGER_PRINT("Error on RT%d: Freetimer= %u, Type= %d (%s),
Bus= %s, SA= %d, ServiceWords= %d, DataWords= %d\n", index, freeTimer,
typeTransaction, getMilTransactionString(typeTransaction), (activeBus ==
MIL1553_BUS_A) ? "A" : "Б", subaddress, countServiceWords16, countDataWords16);
                        printRtError(block);
                        if (typeTransaction == 1) {
                            if ((countDataWords16 > 0) && (isSuccessTransaction))
{
                                unsigned short* dataWord16 = (unsigned
short*)mil1553ud_dmaBlock_getPtrDataPart(block);
                                LOGGER_PRINT("%s: ", "Data words");
                                for(unsigned int i = 0; i < countDataWords16; i++)
{
                                    if (i == countDataWords16-1) {
                                        LOGGER_PRINT("0x%x; \n", dataWord16[i]);
                                    } else {
                                        LOGGER_PRINT("0x%x,", dataWord16[i]);
                                    }
                                }
                            }
                        }
                    }
                    if (MIL1553UD_FALSE ==
isGoodData(&(container.mBlocks[indexBlk]), index)) {
                        PRINT_LINE();
                        LOGGER_PRINT("Error in data on RT%d: Freetimer= %u, Type=
%d, Bus= %s, SA= %d, ServiceWords= %d, DataWords= %d\n", index, freeTimer,
typeTransaction, (activeBus == MIL1553_BUS_A) ? "A" : "Б", subaddress,
countServiceWords16, countDataWords16);
```

```
                        if (typeTransaction == 1) {
                            if ((countDataWords16 > 0) && (isSuccessTransaction))
{
                                unsigned short* dataWord16 = (unsigned
short*)mil1553ud_dmaBlock_getPtrDataPart(block);
                                LOGGER_PRINT("%s: ", "Data words");
                                for(unsigned int i = 0; i < countDataWords16; i++)
{
                                    if (i == countDataWords16-1) {
                                        LOGGER_PRINT("0x%x; \n", dataWord16[i]);
                                    } else {
                                        LOGGER_PRINT("0x%x,", dataWord16[i]);
                                    }
                                }
                            }
                        }
                    }
                } else {
                    LOGGER_PRINT("ReadDma error in RT (%s) = %d\n",
params.mNames[index], ret);
                }
                mil1553_close(fd);
            }
        }
}

void readDevices(void) {
    int index;
    // построение массива обнаруженных девайсов
    mil1553_getArrayDevices(&devices, &devicesCount);
    for (index = 0; index < devicesCount; index++) {
        LOGGER_PRINT("Device: '%s' deviceId= %d, revisionId= %d\n",
devices[index].mDev.mNameChannel, devices[index].mInfo.device_id,
devices[index].mInfo.revision);
    }
}

void readChannels(void) {
    int index;
    // построение массива обнаруженных каналов
    mil1553_getArrayChannels(&channels, &channelsCount);
    for (index = 0; index < channelsCount; index++) {
        int numDev = mil1553_getNumberDevice(&(channels[index]));
        LOGGER_PRINT("Device= %d - Channel: '%s'\n", numDev,
channels[index].mNameChannel);
    }
}

void printResults(void) {
    PRINT_LINE();
    LOGGER_PRINT_STR("Test has finished\n");
    if (TRUE_VAL == configChannels.mIsReady[0])
        LOGGER_PRINT("bc %s: dmablk= %d; err= %d;\n",  params.mNames[0],
statTest.mCh[0].mDmaBlk, statTest.mCh[0].mErr);
    if (TRUE_VAL == configChannels.mIsReady[1])
        LOGGER_PRINT("rt1 %s: dmablk= %d; err= %d;\n", params.mNames[1],
statTest.mCh[1].mDmaBlk, statTest.mCh[1].mErr);
    if (TRUE_VAL == configChannels.mIsReady[2])
        LOGGER_PRINT("rt2 %s: dmablk= %d; err= %d;\n", params.mNames[2],
statTest.mCh[2].mDmaBlk, statTest.mCh[2].mErr);
```

```
    if (TRUE_VAL == configChannels.mIsReady[3])
        LOGGER_PRINT("rt3 %s: dmablk= %d; err= %d;\n", params.mNames[3],
statTest.mCh[3].mDmaBlk, statTest.mCh[3].mErr);
    if (TRUE_VAL == configChannels.mIsReady[4])
        LOGGER_PRINT("rt4 %s: dmablk= %d; err= %d;\n", params.mNames[4],
statTest.mCh[4].mDmaBlk, statTest.mCh[4].mErr);
    if (TRUE_VAL == configChannels.mIsReady[5])
        LOGGER_PRINT("rt5 %s: dmablk= %d; err= %d;\n", params.mNames[5],
statTest.mCh[5].mDmaBlk, statTest.mCh[5].mErr);
    if (TRUE_VAL == configChannels.mIsReady[6])
        LOGGER_PRINT("rt6 %s: dmablk= %d; err= %d;\n", params.mNames[6],
statTest.mCh[6].mDmaBlk, statTest.mCh[6].mErr);
    if (TRUE_VAL == configChannels.mIsReady[7])
        LOGGER_PRINT("rt7 %s: dmablk= %d; err= %d;\n", params.mNames[7],
statTest.mCh[7].mDmaBlk, statTest.mCh[7].mErr);
    if (TRUE_VAL == configChannels.mIsReady[8])
        LOGGER_PRINT("rt8 %s: dmablk= %d; err= %d;\n", params.mNames[8],
statTest.mCh[8].mDmaBlk, statTest.mCh[8].mErr);
}

int isValidNameChannels(void) {
    int ret = TRUE_VAL;
    int index, indexOfAllCh;
    for (index = 0; index < COUNT_CH; index++) {
        if (configChannels.mIsReady[index]) {
            int isGood = FALSE_VAL;
            for (indexOfAllCh = 0; indexOfAllCh < channelsCount; indexOfAllCh++) {
                if (0 == strncmp (params.mNames[index],
channels[indexOfAllCh].mNameChannel, CHANNEL_NAME_LENGTH)) {
                    isGood = TRUE_VAL;
                    break;
                }
            }
            if (isGood == FALSE_VAL) {
                LOGGER_PRINT("Bad channel name in index = %d, name = '%s'\n",
index, params.mNames[index]);
                ret = FALSE_VAL;
            }
        }
    }
    return ret;
}

void bcDebugProcessing(void) {
    int index = 0; // инрдекс КШ
    if (configChannels.mIsReady[index]) {
        SADDR_DATA regBcCondCode, regInstrPtr, regOpDataPtr;
        int fd = mil1553_open(params.mNames[index]);
        unsigned int numChannel = mil1553ud_getNumberChannel(fd);
        regBcCondCode.daddr = MIL_REG(numChannel, GPF_BC_COND_CODE_CH);
        regInstrPtr.daddr = MIL_REG(numChannel, START_ADR_INSTR_PTR_CH);
        regOpDataPtr.daddr = MIL_REG(numChannel, OP_DATA_RAM_PTR_CH);
        mil1553_readReg(fd, &regBcCondCode);
        mil1553_readReg(fd, &regInstrPtr);
        mil1553_readReg(fd, &regOpDataPtr);
        LOGGER_PRINT("> REG_BC_COND_CODE: bcrun= %s, fifoerr= %s, operr= %s,
insterr= %s, mbce= %s\n",
                     ((regBcCondCode.data & (1u<<31)) ? "1" : "0" ),
                     ((regBcCondCode.data & (1u<<3)) ? "1" : "0" ),
                     ( (regBcCondCode.data & (1u<<2)) ? "1" : "0" ),
                     ( (regBcCondCode.data & (1u<<1)) ? "1" : "0" ),
```

```
                    ( (regBcCondCode.data & (1u<<0)) ? "1" : "0" ));
        SADDR_DATA instr;
        instr.daddr = MIL_REG(numChannel, BC_INSTR_RAM_CH) + ((regInstrPtr.data) &
0xFFF)*4;
        mil1553_readReg(fd, &instr);
        LOGGER_PRINT("  REG_INSTR_PTR: sinstr_ptr= 0x%x, cinstr_ptr= 0x%x,
instruction= 0x%x\n",
                     (regInstrPtr.data>>16) & 0xFFF ,
                    (regInstrPtr.data & 0xFFF ) ,
                    (instr.data, 16));
        SADDR_DATA oper;
        oper.daddr = MIL_REG(numChannel, BC_OPERATION_RAM_CH) +
((regOpDataPtr.data>>16) & 0xFFFF)*4;
        mil1553_readReg(fd, &oper);
        LOGGER_PRINT("  REG_OP_DATA_PTR: cop_ptr= 0x%x, cdat_ptr= 0x%x, operation=
0x%x\n",
                     (regOpDataPtr.data>>16) & 0xFFF ,
                    (regOpDataPtr.data & 0xFFF ) ,
                    (oper.data, 16));

        mil1553_close(fd);
    }
}

#define STEP_SECONDS(secstep, seccounter, tvPrev, tvCur, tvDiff) \
    gettimeofday(&tvCur, NULL); \
    timersub(&tvCur, &tvPrev, &tvDiff); \
    if (tvDiff.tv_sec * 1000000 + tvDiff.tv_usec >= (secstep /*seconds*/ *
1000000)) { \
        tvPrev = tvCur; \
        seccounter++; \
        hasStep = MIL1553UD_TRUE; \
    }

void procCTest1(struct TParamCTest1 params) {
    int isValid;
    int countDevices;
    struct timeval tvPrev, tvCur, tvDiff;
    unsigned int allDurationSeconds;
    unsigned int curDurationSeconds;
    gettimeofday(&tvPrev, NULL);
    countDevices = mil1553_getCountDevices();
    if (countDevices == 0) { // проверяем, что девайсов нет - значит и теста нет
        LOGGER_PRINT_STR("Has founded 0 devices. Unable to perform the test.\n");
        goto END;
    }
    readDevices(); // печатаем список всех девайсов
    readChannels(); // печатаем список всех каналов
    isValid = isValidNameChannels(); // проверяем валидность имён каналов в
параметрах
    if (isValid == FALSE_VAL) { // если имена не валидны - теста нет
        LOGGER_PRINT_STR("Channel names is not valid.\n");
        goto END;
    }
    loadMemFiles(); // читаем и парсим мем файлы
    initTestDuration(&params, &allDurationSeconds, &curDurationSeconds); //
устанавливаем длительность теста из параметров
    stopBc(); // останавливаем КШ, если вдруг он с прошлого раза не остановлен
    deinitChannels(); // сброс всех каналов
    initChannels(); // инициализация заданных каналов
    LOGGER_PRINT_STR("Wait for test has finished, please...\n");
```

```
    startBc(); // старт КШ
    isBcStopped = MIL1553UD_FALSE;
    int hasStep = MIL1553UD_TRUE;
    do
    { // цикл теста
        if (MIL1553UD_TRUE == hasStep) { // раз в шаг печатаем отладочную
информацию - это на случай если всё плохо
            //bcDebugProcessing();
            hasStep = MIL1553UD_FALSE;
        }
        processingBc(); // обработка блоков ДМА КШ
        processingRts(); // обработка блоков ДМА ОУ
        STEP_SECONDS(1, curDurationSeconds, tvPrev, tvCur, tvDiff); // отсчитываем
по одной секунде
    }
    while (curDurationSeconds < allDurationSeconds); // проверка завершения теста
по длительности
    stopBc(); // останов КШ
    isBcStopped = MIL1553UD_TRUE;
    unsigned int allDurationSecondsWait = 5;
    unsigned int curDurationSecondsWait = 0;
    if (configChannels.mIsReady[0]) { // если КШ имеется
        gettimeofday(&tvPrev, NULL);
        do { // ждём 5 секунд для остаточной отработки программы КШ
            processingBc(); // обработка блоков ДМА КШ
            processingRts(); // обработка блоков ДМА ОУ
            STEP_SECONDS(1, curDurationSecondsWait, tvPrev, tvCur, tvDiff);
        } while (curDurationSecondsWait < allDurationSecondsWait);
    }
    deinitChannels(); // сброс всех каналов
    printResults(); // печать результатов теста
    PRINT_LINE();
    LOGGER_PRINT("Time spent on the test = %d seconds\n", allDurationSeconds);
END:
    return;
}
```

ПРИЛОЖЕНИЕ В (ИНФОРМАЦИОННОЕ)

Пример визуализации тестовой программы

```
ctest1 - version 29.04.2021
parsing './config/ctest1.txt':
duration_h 8           //продолжительность теста часы
duration_m 5           //продолжительность теста минуты
line 0                  //линия: 0-A; 1-B
bc mil1553dev-0-ch-1     //КШ
rt1 mil1553dev-0-ch-0    //ОУ1
rt2 mil1553dev-0-ch-2    //ОУ2
rt3 mil1553dev-0-ch-3    //ОУ3
rt4 mil1553dev-1-ch-0    //ОУ4
rt5 mil1553dev-1-ch-1    //ОУ5
rt6 mil1553dev-1-ch-2    //ОУ6
rt7 NULL    //ОУ7
rt8 NULL    //ОУ8

PARAMS: duration_h= 8, duration_m= 5, line= 0, bc= /dev/mil1553dev-0-ch-1,
 rt1= /dev/mil1553dev-0-ch-0,
 rt2= /dev/mil1553dev-0-ch-2,
 rt3= /dev/mil1553dev-0-ch-3,
 rt4= /dev/mil1553dev-1-ch-0,
 rt5= /dev/mil1553dev-1-ch-1,
 rt6= /dev/mil1553dev-1-ch-2,
 rt7= NULL,
 rt8= NULL

--- --- ---
Device: '/dev/mil1553dev-7-ch-0' deviceId= 38002, revisionId= 3
Device: '/dev/mil1553dev-6-ch-0' deviceId= 38002, revisionId= 3
Device: '/dev/mil1553dev-5-ch-0' deviceId= 38002, revisionId= 3
Device: '/dev/mil1553dev-4-ch-0' deviceId= 38002, revisionId= 3
Device: '/dev/mil1553dev-3-ch-0' deviceId= 38002, revisionId= 3
Device: '/dev/mil1553dev-2-ch-0' deviceId= 38002, revisionId= 3
Device: '/dev/mil1553dev-1-ch-0' deviceId= 38002, revisionId= 3
Device: '/dev/mil1553dev-0-ch-0' deviceId= 38002, revisionId= 3
Device= 7 - Channel: '/dev/mil1553dev-7-ch-3'
Device= 7 - Channel: '/dev/mil1553dev-7-ch-2'
Device= 7 - Channel: '/dev/mil1553dev-7-ch-1'
Device= 7 - Channel: '/dev/mil1553dev-7-ch-0'
Device= 6 - Channel: '/dev/mil1553dev-6-ch-3'
Device= 6 - Channel: '/dev/mil1553dev-6-ch-2'
Device= 6 - Channel: '/dev/mil1553dev-6-ch-1'
Device= 6 - Channel: '/dev/mil1553dev-6-ch-0'
Device= 5 - Channel: '/dev/mil1553dev-5-ch-3'
Device= 5 - Channel: '/dev/mil1553dev-5-ch-2'
Device= 5 - Channel: '/dev/mil1553dev-5-ch-1'
Device= 5 - Channel: '/dev/mil1553dev-5-ch-0'
Device= 4 - Channel: '/dev/mil1553dev-4-ch-3'
Device= 4 - Channel: '/dev/mil1553dev-4-ch-2'
Device= 4 - Channel: '/dev/mil1553dev-4-ch-1'
Device= 4 - Channel: '/dev/mil1553dev-4-ch-0'
Device= 3 - Channel: '/dev/mil1553dev-3-ch-3'
Device= 3 - Channel: '/dev/mil1553dev-3-ch-2'
Device= 3 - Channel: '/dev/mil1553dev-3-ch-1'
Device= 3 - Channel: '/dev/mil1553dev-3-ch-0'
Device= 2 - Channel: '/dev/mil1553dev-2-ch-3'
Device= 2 - Channel: '/dev/mil1553dev-2-ch-2'
Device= 2 - Channel: '/dev/mil1553dev-2-ch-1'
Device= 2 - Channel: '/dev/mil1553dev-2-ch-0'
Device= 1 - Channel: '/dev/mil1553dev-1-ch-3'
Device= 1 - Channel: '/dev/mil1553dev-1-ch-2'
Device= 1 - Channel: '/dev/mil1553dev-1-ch-1'
Device= 1 - Channel: '/dev/mil1553dev-1-ch-0'
Device= 0 - Channel: '/dev/mil1553dev-0-ch-3'
Device= 0 - Channel: '/dev/mil1553dev-0-ch-2'
Device= 0 - Channel: '/dev/mil1553dev-0-ch-1'
Device= 0 - Channel: '/dev/mil1553dev-0-ch-0'
parsing './mem/consol_dat_ram.mem':
```

```
 - End parsing!
parsing './mem/consol_inst_ram.mem':
 - End parsing!
parsing './mem/consol_op_ram.mem':
 - End parsing!
./mem/consol_inst_ram.mem was writed.
./mem/consol_op_ram.mem was writed.
./mem/consol_dat_ram.mem was writed.
Wait for test has finished, please...
--- --- ---
Test has finished
bc /dev/mil1553dev-0-ch-1: dmablk= 142316289; err= 0;
rt1 /dev/mil1553dev-0-ch-0: dmablk= 29961324; err= 0;
rt2 /dev/mil1553dev-0-ch-2: dmablk= 29961324; err= 0;
rt3 /dev/mil1553dev-0-ch-3: dmablk= 29961324; err= 0;
rt4 /dev/mil1553dev-1-ch-0: dmablk= 29961324; err= 0;
rt5 /dev/mil1553dev-1-ch-1: dmablk= 29961324; err= 0;
rt6 /dev/mil1553dev-1-ch-2: dmablk= 29961324; err= 0;
rt7 NULL: dmablk= 0; err= 0;
rt8 NULL: dmablk= 0; err= 0;
--- --- ---
Time spent on the test = 29100 seconds
```

Рисунок В1 - Пример визуализации тестовой программы

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.*	*№ докумен-та*	*Входящий № сопроводи-тельного докум. и дата*	*Подп.*	*Да-та*
Изм.	*изменен-ных*	*заменен-ных*	*новых*	*аннули-рован-ных*					