

УТВЕРЖДАЮ

Генеральный директор

ООО «НОВОМАР»

\_\_\_\_\_ Т.В. Буга

«\_\_\_»\_\_\_\_\_2023 г.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«БИБЛИОТЕКА СИСТЕМНОГО ВЗАИМОДЕЙСТВИЯ MIL1553UD»

ВЕРСИЙ 3.X

Модулей

“PCIe-1553UDx”

“ХМС-1553UDx”

“CPCIS-1553UDx”

“mPCIe-1553UDx”

“LPCIe-1553UDx”

“USB-1553UDx”

(ОС LINUX)

Руководство программиста

ЛИСТ УТВЕРЖДЕНИЯ

RU.MCKЮ.20105-01 33 01-ЛУ

От

Инженер-программист

\_\_\_\_\_

«\_\_\_»\_\_\_\_\_2023 г.

\_\_\_\_\_

«\_\_\_»\_\_\_\_\_2023 г.

2023

Из	Под	Да

Литера

Ине. №	Подп. и
Взам. инв.	Подп. и
Ине. №	Подп. и

Утвержден  
RU.МСКЮ.20105-01 33 01-ЛУ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ  
«БИБЛИОТЕКА СИСТЕМНОГО ВЗАИМОДЕЙСТВИЯ MIL1553UD»  
ВЕРСИЙ 3.X

Модулей  
“PCIe-1553UDx”  
“ХМС-1553UDx”  
“СРСIS-1553UDx”  
“mPCIe-1553UDx”  
“LPCIe-1553UDx”  
“USB-1553UDx”

**(ОС LINUX)**  
Руководство программиста

RU.МСКЮ.20105-01 33 01

Листов 59

2023

Изд	Под	Да

Литера

Инв. №	Подп. и	Взам. инв.	Инв. №	Подп. и

## АННОТАЦИЯ

В книге описываются технологические принципы, использованные в программном обеспечении «БИБЛИОТЕКА СИСТЕМНОГО ВЗАИМОДЕЙСТВИЯ MIL1553UD» (ОС Linux и Astra Linux) версий 3.x, для работы модулей: «PCIE-1553UDx», «ХМС-1553UDx», «СРСIS-1553UDx», «mPCIE-1553UDx», «LPCIE-1553UDx», «USB-1553UDx» в сети МКИО ГОСТ Р 52070-2003. В частности, рассмотрены функциональное назначение и область применения, условия выполнения.

Из	Под	Да

## СОДЕРЖАНИЕ

1	НАЗНАЧЕНИЕ ПРОГРАММЫ .....	6
2	УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ .....	7
3	ХАРАКТЕРИСТИКА ПРОГРАММЫ .....	8
4	ОБРАЩЕНИЕ К ПРОГРАММЕ.....	9
4.1	МЕТОДЫ БИБЛИОТЕКИ СИСТЕМНОГО ВЗАИМОДЕЙСТВИЯ MIL1553UD.....	10
4.1.1	Получить номер канала из символьного устройства канала.....	10
4.1.2	Генерация имени регистра в соответствии с номером канала.....	10
4.1.3	Получить количество блоков DMA, готовых для чтения.....	10
4.1.4	Создать контейнер блоков DMA .....	11
4.1.5	Освобождает контейнер блоков DMA.....	11
4.1.6	Считывает все блоки DMA, готовые к чтению, из канала .....	11
4.1.7	Получить количество блоков прерываний, готовых для чтения .....	11
4.1.8	Считывает блоки прерываний, готовые к чтению, из канала.....	12
4.1.9	Управление микропрограммой в режиме КШ .....	12
4.1.10	Установить стартовый адрес инструкции микропрограммы КШ.....	12
4.1.11	Создать контейнер для микропрограммы КШ.....	13
4.1.12	Освободить контейнер для микропрограммы КШ .....	13
4.1.13	Формирование инструкции для микропрограммы КШ .....	13
4.1.14	Формирование нечётной части операции для микропрограммы КШ	14
4.1.15	Формирование чётной части операции для микропрограммы КШ...	14
4.1.16	Формирование командного слова операции для микропрограммы КШ	14
4.1.17	Получить указатель на блок DMA из контейнера.....	15
4.1.18	Проверка на допустимость контейнера DMA.....	15
4.1.19	Получить тип транзакции из блока DMA.....	15
4.1.20	Проверить успешность транзакции.....	15
4.1.21	Получить количество служебных 16-разрядных слов .....	16
4.1.22	Получить количество 16-разрядных слов данных.....	16
4.1.23	Получить активную шину из блока DMA .....	16
4.1.24	Получить указатель на часть с данными блока DMA .....	17

Из	Под	Да

4.1.25	Получить подадрес из блока DMA в режиме ОУ .....	17
4.1.26	Открыть канал (символьное устройство драйвера) на чтение и запись 17	
4.1.27	Открыть канал (символьное устройство драйвера).....	17
4.1.28	Закрыть канал (символьное устройство драйвера).....	18
4.1.29	Чтение блоков из DMA буфера канала (актуальная версия).....	18
4.1.30	Чтение блоков из DMA буфера канала (устаревшая версия) .....	19
4.1.31	Включение/выключение канала .....	20
4.1.32	Выбор роли (режима) канала .....	20
4.1.33	Получить количество каналов со всех девайсов (API драйвера).....	20
4.1.34	Получить количество девайсов (API драйвера).....	20
4.1.35	Получить массив описателей девайсов (API драйвера).....	21
4.1.36	Получить номер девайса, к которому принадлежит канал (API драйвера) .....	21
4.1.37	Получить список каналов всех устройств (API драйвера).....	21
4.1.38	Получить список каналов заданного девайса (API драйвера).....	22
4.1.39	Записать значение в регистр .....	22
4.1.40	Прочитать значение из регистра.....	22
4.1.41	Изменить заданные биты регистра.....	22
4.1.42	Получить информацию о девайсе .....	23
4.1.43	Получить версию драйвера .....	23
4.1.44	Получить информацию о PCI локации девайса (API драйвера) .....	23
4.1.45	Получить информацию о USB адресе девайса (API драйвера).....	23
4.1.46	Сброс DMA .....	24
4.1.47	Управление DMA .....	24
4.1.48	Выбор шины А и/или Б.....	24
4.1.49	Запись данных в RAM КШ.....	25
4.1.50	Разрешение/запрет подадресов ОУ на приём/передачу .....	26
4.1.51	Установить адрес ОУ .....	26
4.1.52	Управление режимом буфера ОУ на передачу .....	26
4.1.53	Управление готовностью буфера ОУ на передачу .....	27
4.1.54	Запись данных в подадрес отправного буфера ОУ .....	27
5	СООБЩЕНИЯ.....	28

Из	Под	Да

5.1.1	Обработка ошибок команд.....	28
	ПРИЛОЖЕНИЕ А (ОБЯЗАТЕЛЬНОЕ) .....	29
	ПРИЛОЖЕНИЕ Б (ИНФОРМАЦИОННОЕ).....	30
	ПРИЛОЖЕНИЕ В (ИНФОРМАЦИОННОЕ) .....	56
	СПИСОК СОКРАЩЕНИЙ.....	58

Из	Под	Да

## 1 НАЗНАЧЕНИЕ ПРОГРАММЫ

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 (API с использованием символьных устройств, предоставляемых драйвером – интерфейсы PCI или USB).

Из	Под	Да

## 2 УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

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

В процессе выполнения функций библиотеки, идут обращения к целевому устройству через интерфейс символьных устройств, предоставляемый драйвером MIL1553UD\_PCI или MIL1553UD\_USB.

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.

Из	Под	Да



### 3 ХАРАКТЕРИСТИКА ПРОГРАММЫ

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

- mil1553ud\_cmd.h
- mil1553ud\_common.h
- mil1553ud\_regs.h
- mil1553ud\_ioctl\_wrappers.h
- mil1553ud\_library.h
- mil1553ud\_library.c

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

Отличия библиотек для ОС Linux и ОС Windows.

Метод получения списка устройств:

- в Linux файловая система видна для пользователя;
- в Windows драйвер возвращает список.

Первый аргумент функций:

- int (Linux-дескриптор);
- структура, которая содержит Windows-дескриптор и номер канала.

Из	Под	Да

#### 4 ОБРАЩЕНИЕ К ПРОГРАММЕ

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

Взаимодействие с каналами целевого устройства происходит с помощью функций, представленных в заголовочном файле «mil1553ud\_library.h», посредством использования API символьных устройств, предоставляемых драйвером MIL1553UD\_PCI или MIL1553UD\_USB.

В качестве первого аргумента, функции библиотеки принимают дескриптор символьного устройства (`int fd`), открыть который необходимо при помощи `mil1553_open`, а для корректного завершения работы — закрыть методом `mil1553_close`.

Пример исходного кода программы-примера приведён в приложении Б.

Скриншоты работы тестовой программы приведены в приложении В.

Из	Под	Да

## 4.1 МЕТОДЫ БИБЛИОТЕКИ СИСТЕМНОГО ВЗАИМОДЕЙСТВИЯ MIL1553UD

### 4.1.1 Получить номер канала из символьного устройства канала

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 1.

```

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

```

Рисунок 1 – Листинг функции

### 4.1.2 Генерация имени регистра в соответствии с номером канала

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 2.

```

///
/// \brief макрос генерирует 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 2 – Листинг функции

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

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 3.

```

///
/// \brief mil1553ud_getCountDmaBlocksReadyRead получить количество блоков
DMA, готовых для чтения
/// \param fd дескриптор файла
/// \param codeError код возврата (признак ошибки)
/// \return кол-во блоков DMA, готовых к чтению
///
unsigned int mil1553ud_getCountDmaBlocksReadyRead(int fd, unsigned int*
codeError);

```

Рисунок 3 – Листинг функции

Из	Под	Да

#### 4.1.4 Создать контейнер блоков DMA

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 4.

```

///
/// \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 4 – Листинг функции

#### 4.1.5 Освобождает контейнер блоков DMA

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 5.

```

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

```

Рисунок 5 – Листинг функции

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

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 6.

```

///
/// \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, unsigned int
*cntBlocks);

```

Рисунок 6 – Листинг функции

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

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 7.

```

///
/// \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 7 – Листинг функции

Из	Под	Да

#### 4.1.8 Считывает блоки прерываний, готовые к чтению, из канала

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 8.

```

///
/// \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 8 – Листинг функции

#### 4.1.9 Управление микропрограммой в режиме КШ

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 9.

```

///
/// \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 9 – Листинг функции

#### 4.1.10 Установить стартовый адрес инструкции микропрограммы КШ

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 10.

```

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

```

Рисунок 10 – Листинг функции

Из	Под	Да

## 4.1.11 Создать контейнер для микропрограммы КШ

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 11.

```

///
/// \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-разрядных (двойных) словах
/// \param shiftInDwords смещение в памяти КШ в 32-разрядных (двойных)
словах
/// \return указатель на контейнер для микропрограммы КШ
///
BC_RAM_BLOCK* mil1553ud_newBcRamBlock(unsigned int type, unsigned int
sizeInDwords, unsigned int shiftInDwords);

```

Рисунок 11 – Листинг функции

## 4.1.12 Освободить контейнер для микропрограммы КШ

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 12.

```

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

```

Рисунок 12 – Листинг функции

## 4.1.13 Формирование инструкции для микропрограммы КШ

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 13.

```

///
/// \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 13 – Листинг функции

Из	Под	Да

## 4.1.14 Формирование нечётной части операции для микропрограммы КШ

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 14.

```

///
/// \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 14 – Листинг функции

## 4.1.15 Формирование чётной части операции для микропрограммы КШ

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 15.

```

///
/// \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 15 – Листинг функции

## 4.1.16 Формирование командного слова операции для микропрограммы КШ

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 16.

```

///
/// \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 16 – Листинг функции

Из	Под	Да

## 4.1.17 Получить указатель на блок DMA из контейнера

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 17.

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

Рисунок 17 – Листинг функции

## 4.1.18 Проверка на допустимость контейнера DMA

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 18.

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

Рисунок 18 – Листинг функции

## 4.1.19 Получить тип транзакции из блока DMA

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 19.

```
///
/// \brief mil1553ud_dmaBlock_getTypeTransaction получить тип транзакции из
блока DMA
/// тип транзакции соответствует 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 19 – Листинг функции

## 4.1.20 Проверить успешность транзакции

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 20.

```
///
/// \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 20 – Листинг функции

Из	Под	Да



## 4.1.21 Получить количество служебных 16-разрядных слов

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 21.

```

///
/// \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 21 – Листинг функции

## 4.1.22 Получить количество 16-разрядных слов данных

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 22.

```

///
/// \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 22 – Листинг функции

## 4.1.23 Получить активную шину из блока DMA

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 23.

```

///
/// \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 23 – Листинг функции

Из	Под	Да

## 4.1.24 Получить указатель на часть с данными блока DMA

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 24.

```

/// \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 24 – Листинг функции

## 4.1.25 Получить подадрес из блока DMA в режиме ОУ

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 25.

```

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

```

Рисунок 25 – Листинг функции

## 4.1.26 Открыть канал (символьное устройство драйвера) на чтение и запись

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 26.

```

/// \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 26 – Листинг функции

## 4.1.27 Открыть канал (символьное устройство драйвера)

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 27.

```

/// \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 27 – Листинг функции

Из	Под	Да

## 4.1.28 Закрывать канал (символьное устройство драйвера)

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 28.

```

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

```

Рисунок 28 – Листинг функции

## 4.1.29 Чтение блоков из ДМА буфера канала (актуальная версия)

Позволяет считать блоки в пользовательский буфер за один вызов, без предварительного запроса доступного количества. В структуре результата также заполняется поле с количеством блоков, оставшихся в буфере устройства.

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 29.

```

#define MIL1553_ONE_BLOCK_SIZE_128_BYTES        128

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

///
/// \brief mil1553_readDmaFixBlocks чтение фиксированное кол-во блоков
/// \param fd дескриптор канала
/// \param container контейнер блоков ДМА
/// \return результат операции
///
unsigned int mil1553_readDmaFixBlocks(int fd, MIL1553_DMA_FIX_BLOCK*
container);

```

Рисунок 29 – Листинг функции

Из	Под	Да

## 4.1.30 Чтение блоков из ДМА буфера канала (устаревшая версия)

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 30.

```

/// \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
/// \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 30 – Листинг функции

Из	Под	Да

## 4.1.31 Включение/выключение канала

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 31.

```

/// \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 31 – Листинг функции

## 4.1.32 Выбор роли (режима) канала

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 32.

```

///
/// \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 32 – Листинг функции

## 4.1.33 Получить количество каналов со всех девайсов (API драйвера)

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 33.

```

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

```

Рисунок 33 – Листинг функции

## 4.1.34 Получить количество девайсов (API драйвера)

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 34.

```

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

```

Рисунок 34 – Листинг функции

Из	Под	Да

## 4.1.35 Получить массив описателей девайсов (API драйвера)

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 35.

```

///
/// \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 35 – Листинг функции

## 4.1.36 Получить номер девайса, к которому принадлежит канал (API драйвера)

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 36.

```

///
/// \brief mil1553_getNumberDevice получить номер устройства, к
которому принадлежит канал
/// \param channel канал
/// \return номер устройства
///
int mil1553_getNumberDevice(struct Tmil1553Channel * channel);

```

Рисунок 36 – Листинг функции

## 4.1.37 Получить список каналов всех устройств (API драйвера)

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 37.

```

///
/// \brief mil1553_getArrayChannels получить список каналов всех
устройств
/// \param channels каналы
/// \param length размер массива
///
void mil1553_getArrayChannels(struct Tmil1553Channel **channels, int
*length);

```

Рисунок 37 – Листинг функции

Из	Под	Да

## 4.1.38 Получить список каналов заданного девайса (API драйвера)

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 38.

```

/// \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 38 – Листинг функции

## 4.1.39 Записать значение в регистр

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 39.

```

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

```

Рисунок 39 – Листинг функции

## 4.1.40 Прочитать значение из регистра

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 40.

```

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

```

Рисунок 40 – Листинг функции

## 4.1.41 Изменить заданные биты регистра

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 41.

```

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

```

Рисунок 41 – Листинг функции

Из	Под	Да

## 4.1.42 Получить информацию о девайсе

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 42.

```

///
/// \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 42 – Листинг функции

## 4.1.43 Получить версию драйвера

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 43.

```

///
/// \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 43 – Листинг функции

## 4.1.44 Получить информацию о PCI локации девайса (API драйвера)

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 44.

```

///
/// \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 44 – Листинг функции

## 4.1.45 Получить информацию о USB адресе девайса (API драйвера)

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 45.

```

///
/// \brief mil1553_getDeviceUSBInfo получить информацию о настроенном USB
адресе платы
/// \param fd дескриптор канала
/// \param usbInfo информация об адресе
/// \return результат операции
///
unsigned int mil1553_getDeviceUsbInfo(int fd, USB_LOCATION* usbInfo);

```

Рисунок 45 – Листинг функции

Из	Под	Да



#### 4.1.46 Сброс ДМА

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 46.

```

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

```

Рисунок 46 – Листинг функции

#### 4.1.47 Управление ДМА

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 47.

```

///
/// \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 47 – Листинг функции

#### 4.1.48 Выбор шины А и/или Б

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 48.

```

///
/// \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 48 – Листинг функции

Из	Под	Да

## 4.1.49 Запись данных в RAM КШ

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 49.

```

///
/// \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 49 – Листинг функции

Из	Под	Да

## 4.1.50 Разрешение/запрет подадресов ОУ на приём/передачу

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 50.

```

///
/// \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 подадрес) (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 50 – Листинг функции

## 4.1.51 Установить адрес ОУ

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 51.

```

///
/// \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 51 – Листинг функции

## 4.1.52 Управление режимом буфера ОУ на передачу

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 52.

```

///
/// \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 52 – Листинг функции

Из	Под	Да

## 4.1.53 Управление готовностью буфера ОУ на передачу

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 53.

```

///
/// \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 53 – Листинг функции

## 4.1.54 Запись данных в подадрес отправного буфера ОУ

Описание функции в файле «mil1553ud\_library.h» приведено на рисунке 54.

```

/// \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 54 – Листинг функции

Из	Под	Да

## СООБЩЕНИЯ

В процессе работы с библиотекой системного взаимодействия, драйвер MIL1553 сохраняет отладочную информацию, которую можно увидеть с помощью команды `dmesg`, набранную в терминале ОС Linux.

## 5.1.1 Обработка ошибок команд

Описание кодов ошибок приведено на рисунке 125.

```

/// \brief невалидное значение, свидетельствующее об ошибке
#define LIB_BAD_VALUE          0xFFFFFFFF
/// \brief значение отсутствия ошибки
#define LIB_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 125 – Листинг

Из	Под	Да

## 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".

Настройка количества и параметров подключенных устройств «USB-1553UDx» осуществляется с помощью утилиты «КОНФИГУРАТОР АВТОЗАГРУЗКИ USB-1553UDX»

Из	Под	Да

## ПРИЛОЖЕНИЕ Б (ИНФОРМАЦИОННОЕ)

## Листинг программы-примера

## Листинг Б1 – Листинг программы

```

#ifndef _DMA_TEST_H
#define _DMA_TEST_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, участвующих 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
    int mIsRemote[COUNT_CH];  ///< признак удалённо управляемого 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

```

Из	Под	Да

```

///
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];
    int mBcInt;
};
/// \brief неиспользуемй канал
#define NULL_DEV "NULL"
/// \brief удаленно-управляемй канал
#define REMOTE_DEV "REMOTE"
#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"
///< показывать прерывания кш
#define T_CTEST1_BCINT "bcint"

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;    ///< конфигурация 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[]);

```

Из	Под	Да



```

///
/// \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 // _DMA_TEST_H

#include "dma_test.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
#include <iterator>
#include <string>
#include <regex>

char configFileName[MAX_LEN_CONFIG_FILE];
char fullConfigFileName[MAX_LEN_FULL_CONFIG_FILE];
struct TParamCTest1 params;
struct TMill153Channel* channels;
int channelsCount;
struct TMill153Device* devices;
int devicesCount;
struct SConfigCh configChannels; ///< конфигурация 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

```

Из	Под	Да

```

typedef struct
{
    char name[ CHANNEL_NAME_LENGTH ];
    bool isExt;
    int fd;
    Mill1553ChannelFD extFd;
} Abonent;

#ifdef _WIN32

/**
 \brief обёртка для точно такого же вызова, но с приставкой ext, для внешних устройств
 \param abonent структура "абонент"
 \param FUNC название команды для выполнения, после которого следуют аргументы, передаваемые команде
 */
#define IS_EXT_WRAPPER(abonent, FUNC, ...) \
    ext##FUNC( abonent.extFd, ##__VA_ARGS__ )

#else

/**
 \brief обёртка для точно такого же вызова, но с приставкой ext, для внешних устройств
 \param abonent структура "абонент"
 \param FUNC название команды для выполнения, после которого следуют аргументы, передаваемые команде
 */
#define IS_EXT_WRAPPER(abonent, FUNC, ...) \
    (abonent.isExt) ? ext##FUNC( abonent.extFd, ##__VA_ARGS__ ) : FUNC(
abonent.fd, ##__VA_ARGS__ )

#endif // _WIN32

struct AbonentItem
{
    Abonent abonent;
    struct AbonentItem *next;
};

struct AbonentItem *g_abonentCache = NULL;

void close_all_abonents()
{
    struct AbonentItem *tmp;

    while(g_abonentCache != NULL)
    {
        if(g_abonentCache->abonent.isExt)
            extmill1553_close(g_abonentCache->abonent.extFd);
#ifdef _WIN32
        else
            mill1553_close(g_abonentCache->abonent.fd);
#endif

        tmp = g_abonentCache->next;
        free(g_abonentCache);
    }
}

```

Из	Под	Да

```

        g_abonentCache = tmp;
    }
}

Abonent open_abonent_from_string(char *name)
{
    std::string nameStr(name);
    Abonent result;

    for(struct AbonentItem *abonentCacheTmp = g_abonentCache; abonentCacheTmp !=
    NULL; abonentCacheTmp = abonentCacheTmp->next)
        if(0 == strcmp(name, abonentCacheTmp->abonent.name, CHANNEL_NAME_LENGTH))
            return abonentCacheTmp->abonent;

    memset(&result, 0, sizeof(Abonent));
    result.isExt = false;
    result.fd = -1;
    result.extFd.socketFD = -1;
    result.extFd.channel = 0;
    strncpy(result.name, name, CHANNEL_NAME_LENGTH);

    std::string externPathStr("("
                                "(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9][0-9]|[0-
    9])\\\\"
                                "(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9][0-9]|[0-
    9])\\\\"
                                "(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9][0-9]|[0-
    9])\\\\"
                                "(25[0-5]|2[0-4][0-9]|1[0-9][0-9]|[1-9][0-9]|[0-
    9])"
                                "):([0-9]{1,5}):([0-9]{1,2})$");

    /* формат внешнего пути: ip-адрес:порт:канал */
    std::regex externPathRegex(externPathStr, std::regex::extended);

    std::smatch matches;
    std::regex_search(nameStr, matches, externPathRegex);

    /* выполнение регулярного выражения */
    if (matches.size() == 8)
    {
        std::string str_ipAddr, str_port, str_channel;

        str_ipAddr = matches[1];
        str_port = matches[6];
        str_channel = matches[7];

        int port, channel;

        channel = atoi(str_channel.c_str());
        port = atoi(str_port.c_str());

        result.isExt = true;
        result.extFd = extmill553_open(str_ipAddr.c_str(), port, channel);
    }
    else
    {
#ifdef _WIN32
        abort();
#else
        // локальное устройство

```

Из	Под	Да

```

result.fd = mill1553_open(name);
result.extFd.channel = mill1553ud_getNumberChannel(result.fd);
#endif
}

{
    struct AbonentItem *nextAbonentItem;

    if(g_abonentCache == NULL)
    {
        g_abonentCache = (struct AbonentItem *) malloc(sizeof(struct
AbonentItem));
        nextAbonentItem = g_abonentCache;
    }
    else
    {
        nextAbonentItem = g_abonentCache;

        while(nextAbonentItem->next != NULL)
            nextAbonentItem = nextAbonentItem->next;

        nextAbonentItem->next = (struct AbonentItem *) malloc(sizeof(struct
AbonentItem));
        nextAbonentItem = nextAbonentItem->next;
    }

    memset(nextAbonentItem, 0, sizeof(struct AbonentItem));
    nextAbonentItem->abonent = result;
}

return result;
}

void generateDataRt(Abonent abonent);

int main(int argc, char* argv[]) {
#ifdef _WIN32
    setlocale(LC_CTYPE, "rus"); // вызов функции настройки локали
#endif

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

```

Из	Под	Да

```

        LOGGER_PRINT("Невозможно прочитать параметры теста из '%s' \n",
                    fullConfigFileName);
    } else { //параметры из конфигурационного файла - ок
        procCTest1(params);
    }
}
}
LOGGER_FLUSH();
LOGGER_CLOSE();

close_all_abonents();

return ret;
}
// \brief размер мем файла
#define MEM_SIZE 1024
// \brief инструкции КММ
static unsigned int memDesc[MEM_SIZE];
// \brief операции КММ
static unsigned int memOp[MEM_SIZE];
// \brief данные КММ
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]) &&
        (FALSE_VAL == configChannels.mIsRemote[num_rt]))
    {
        unsigned int adr = cycleAddresses[counter] + num_rt - 1;
        memDescRtOff(&(memDesc[adr]));
    }
}

```

Из	Под	Да

```

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;
        Abonent abonent = open_abonent_from_string(params.mNames[0]);

        isGood = IS_EXT_WRAPPER(abonent, mill1553_bc_writeDataToRam,
Mill1553BcTypeInstructions, memDesc, MEM_SIZE);
        LOGGER_PRINT_STR("./mem/consol_inst_ram.mem");
        if (isGood == GOOD_CODE)
            LOGGER_PRINT_STR(" записан.\n");
        else
            LOGGER_PRINT(" НЕ записан. Ошибка %d\n", isGood);

        isGood = IS_EXT_WRAPPER(abonent, mill1553_bc_writeDataToRam,
Mill1553BcTypeOperations, memOp, MEM_SIZE);
        LOGGER_PRINT_STR("./mem/consol_op_ram.mem");
        if (isGood == GOOD_CODE)
            LOGGER_PRINT_STR(" записан.\n");
        else
            LOGGER_PRINT(" НЕ записан. Ошибка %d\n", isGood);

        isGood = IS_EXT_WRAPPER(abonent, mill1553_bc_writeDataToRam,
Mill1553BcTypeData, memData, MEM_SIZE);
        LOGGER_PRINT_STR("./mem/consol_dat_ram.mem");
    }
}

```

Из	Под	Да

```

        if (isGood == GOOD_CODE)
            LOGGER_PRINT_STR(" записан.\n");
        else
            LOGGER_PRINT(" НЕ записан. Ошибка %d\n", isGood);
    }
}

void initChannel_switchOffChannel(Abonent abonent)
{
    int isGood;
    unsigned int opCh = MIL_DEV_CHANNEL_OFF;
    isGood = IS_EXT_WRAPPER(abonent, mill1553_manChannel, opCh);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mill1553_manChannel\n",
abonent.name);

    unsigned int opDma = MIL1553_OFF;
    isGood = IS_EXT_WRAPPER(abonent, mill1553_manDma, opDma);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mill1553_manDma\n",
abonent.name);

    isGood = IS_EXT_WRAPPER(abonent, mill1553_clearDma);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mill1553_clearDma\n",
abonent.name);
}

void initChannel_config(Abonent abonent, int index)
{
    int isGood;
    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;

    isGood = IS_EXT_WRAPPER(abonent, mill1553_setRole, modeBus);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mill1553_manChannel\n",
abonent.name);

    isGood = IS_EXT_WRAPPER(abonent, mill1553_switchBus, bus);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mill1553_manChannel\n",
abonent.name);

    if (0 != index)
    {
        isGood = IS_EXT_WRAPPER(abonent, mill1553_rt_setAddress, index);
        if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mill1553_manChannel\n",
abonent.name);
    }
}

void initChannel_initRtSubaddresses(Abonent abonent)
{
    int isGood;

    isGood = IS_EXT_WRAPPER(abonent, mill1553_rt_manSubaddresses,
MIL_RT_BUF_RECEIVE, 0x7FFFFFFE /* меняем с 1 по 30
подадрес*/,
0x7FFFFFFE /* включаем на приём с 1 по 30 подадрес*/);
}

```

Из	Под	Да

```

    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mill1553_rt_manSubaddresses
MIL_RT_BUF_RECEIVE\n", abonent.name);

    isGood = IS_EXT_WRAPPER(abonent, mill1553_rt_manSubaddresses,
MIL_RT_BUF_TRANSMIT, 0x7FFFFFFE /* меняем с 1 по 30
подадрес*/,
                                0x7FFFFFFE /* включаем на приём с 1 по 30 подадрес*/);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mill1553_rt_manSubaddresses
MIL_RT_BUF_TRANSMIT\n", abonent.name);
}

void initChannel_initRtSubaddressBuffer(Abonent abonent)
{
    int isGood;
    unsigned char subaddress;

    for (subaddress = 1; subaddress <= 30; subaddress++)
    {
        isGood = IS_EXT_WRAPPER(abonent,
mill1553_rt_manSubadressTransmitBufferMode, subaddress, MIL_RT_MODE_PROG);
        if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка
mill1553_rt_manSubadressTransmitBufferMode\n", abonent.name);

        isGood = IS_EXT_WRAPPER(abonent,
mill1553_rt_manSubadressTransmitBufferReady, subaddress, MIL_RT_BUF_0_ON_1_OFF);
        if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка
mill1553_rt_manSubadressTransmitBufferReady\n", abonent.name);
    }

    generateDataRt(abonent);
}

void initChannel_switchOnChannel(Abonent abonent)
{
    int isGood;

    // включаем ДМА
    unsigned int opDma = MIL1553_ON;
    isGood = IS_EXT_WRAPPER(abonent, mill1553_manDma, opDma);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mill1553_manDma MIL1553_ON\n",
abonent.name);

    // включаем канал
    unsigned int opCh = MIL_DEV_CHANNEL_ON;
    isGood = IS_EXT_WRAPPER(abonent, mill1553_manChannel, opCh);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mill1553_manChannel
MIL_DEV_CHANNEL_ON\n", abonent.name);
}

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])
        {
            Abonent abonent = open_abonent_from_string(params.mNames[index]);

            if (0 == index)
            {

```

Из	Под	Да



```

        printf("%s: INIT BC\n", abonent.name);
    }
    else
    {
        printf("%s: INIT RT%d\n", abonent.name, index);
    }

    printf("%s: initChannel_switchOffChannel\n", abonent.name);
    initChannel_switchOffChannel(abonent); // выкл канала
    printf("%s: initChannel_config\n", abonent.name);
    initChannel_config(abonent, index); // конфигурация канала
    if (0 != index) { // для ОУ
        printf("%s: initChannel_initRtSubaddresses\n", abonent.name);
        initChannel_initRtSubaddresses(abonent); // установка разрешений
поадресов
        printf("%s: initChannel_initRtSubaddressBuffer\n", abonent.name);
        initChannel_initRtSubaddressBuffer(abonent); // установка буферов
на передачу
    }
    initChannel_switchOnChannel(abonent); // вкл канала
    printf("%s: --- END INIT\n", abonent.name);
}
}
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 isGood;
            Abonent abonent = open_abonent_from_string(params.mNames[index]);

            // деинициализация
            unsigned int opCh = MIL_DEV_CHANNEL_OFF;
            isGood = IS_EXT_WRAPPER(abonent, mill1553_manChannel, opCh);
            if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mill1553_manChannel
MIL_DEV_CHANNEL_OFF\n", abonent.name);

            unsigned int opDma = MIL1553_OFF;
            isGood = IS_EXT_WRAPPER(abonent, mill1553_manDma, opDma);
            if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mill1553_manDma
MIL1553_OFF\n", abonent.name);

            isGood = IS_EXT_WRAPPER(abonent, mill1553_clearDma);
            if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mill1553_clearDma\n",
abonent.name);
        }
    }
}

void startBc(void)
{
    int isGood;

    if (configChannels.mIsReady[0])
    { // старт канала КШ, если он есть
        Abonent abonent = open_abonent_from_string(params.mNames[0]);

```

Из	Под	Да

```

int numCh = abonent.extFd.channel;

int shift = 0;
switch (numCh)
{
case CH_NUM_1:
    shift = 10;
    break;
case CH_NUM_2:
    shift = 14;
    break;
case CH_NUM_3:
    shift = 18;
    break;
case CH_NUM_4:
    shift = 22;
    break;
default: break;
}

SADDR_DATA reg;
reg.daddr = INTERRUPT_MASK;
reg.data = 1 << shift;

if (params.mBcInt)
{
    isGood = IS_EXT_WRAPPER(abonent, mill1553_writeReg, &reg);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mill1553_writeReg\n",
abonent.name);
}

    isGood = IS_EXT_WRAPPER(abonent, mill1553ud_bcSetStartInstructionAddress,
BC_START_ADDRESS);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка
mill1553ud_bcSetStartInstructionAddress\n", abonent.name);

    reg.daddr = MIL_REG(numCh, START_ADR_INSTR_PTR_CH); // адрес регистра
isGood = IS_EXT_WRAPPER(abonent, mill1553_readReg, &reg);
if(isGood != GOOD_CODE)
    LOGGER_PRINT("%s: Ошибка mill1553_readReg START_ADR_INSTR_PTR_CH\n",
abonent.name);
else
    LOGGER_PRINT("BC READ REG '%s' ADR=0x%lx, VAL=0x%x\n",
"START_ADR_INSTR_PTR_CH", reg.daddr, reg.data);

    isGood = IS_EXT_WRAPPER(abonent, mill1553ud_bcProgram, BC_PROGRAM_START);
// BCSTRT - старт выполнения программы КШ;
if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mill1553ud_bcProgram
BC_PROGRAM_START\n", abonent.name);

    reg.daddr = MIL_REG(numCh, BC_CONF_REG_PCI_CH); // адрес регистра
isGood = IS_EXT_WRAPPER(abonent, mill1553_readReg, &reg);
if(isGood != GOOD_CODE)
    LOGGER_PRINT("%s: Ошибка mill1553_readReg BC_CONF_REG_PCI_CH\n",
abonent.name);
else
    LOGGER_PRINT("BC READ REG '%s' ADR=0x%lx, VAL=0x%x\n",
"BC_CONF_REG_PCI_CH", reg.daddr, reg.data);
}
}

```

Из	Под	Да

```

void stopBc(void)
{
    int isGood;

    if (configChannels.mIsReady[0])
    { // старт канала КШ, если он есть
        Abonent abonent = open_abonent_from_string(params.mNames[0]);

        isGood = IS_EXT_WRAPPER(abonent, mill1553ud_bcProgram, BC_PROGRAM_STOP);
// BCSTART - старт выполнения программы КШ;
        if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mill1553ud_bcProgram
BC_PROGRAM_STOP\n", abonent.name);
    }
}

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

int readConfigFile(char* fullname, struct TParamCTest1* params) {
    char cwdBuf[2500];
    getcwd(cwdBuf, 2500);
    LOGGER_PRINT("Рабочий каталог, откуда запущена программа: '%s'\n", cwdBuf);

    FILE* file = fopen(fullname, "r");
    char* str;
    if (file == NULL) {
        LOGGER_PRINT("Невозможно открыть конфигурационный файл '%s'\n", fullname);
        return BAD_CODE;
    }
    LOGGER_PRINT("парсинг '%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_BCINT, sizeof(T_CTEST1_BCINT))) {
            sscanf(str, "%s %d", typeArgStr, &valArg);
            params->mBcInt = valArg;
        }
        else
        {
            int num;
            static const char *names[] = {
                T_CTEST1_BC,

```

Из	Под	Да

```

T_CTEST1_RT1,
T_CTEST1_RT2,
T_CTEST1_RT3,
T_CTEST1_RT4,
T_CTEST1_RT5,
T_CTEST1_RT6,
T_CTEST1_RT7,
T_CTEST1_RT8,
NULL
};

for(num = 0; names[ num ] != NULL; ++num)
{
    if(0 == strcmp(typeArgStr, names[ num ]))
    {
        sscanf(str, "%s %s", typeArgStr, valStr);
        configChannels.mIsRemote[num] = FALSE_VAL;
        if (0 == strncmp(valStr, NULL_DEV, sizeof(NULL_DEV)))
        {
            configChannels.mIsReady[num] = FALSE_VAL;
            memcpy(params->mNames[num], valStr, CHANNEL_NAME_LENGTH);
        }
        else if (0 == strncmp(valStr, REMOTE_DEV, sizeof(REMOTE_DEV)))
        {
            configChannels.mIsReady[num] = FALSE_VAL;
            configChannels.mIsRemote[num] = TRUE_VAL;
            memcpy(params->mNames[num], valStr, CHANNEL_NAME_LENGTH);
        }
        else
        {
            char fullNameCh[CHANNEL_NAME_LENGTH];
            configChannels.mIsReady[num] = TRUE_VAL;
            sprintf(fullNameCh, CHANNEL_NAME_LENGTH, "%s%s", STR_DEV,
valStr);
            memcpy(params->mNames[num], fullNameCh,
CHANNEL_NAME_LENGTH);
        }
        break;
    }
}
memset(typeArgStr, 0, 30);
}
fclose(file);
LOGGER_PRINT(
    "ПАРАМЕТРЫ: 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 bcint= %d \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], params->
>mBcInt);
    PRINT_LINE();
    return GOOD_CODE;
}

void initLogCTest1(void) {

```

Из	Под	Да

```

char filename[LOGGER_FILENAME_LENGTH];

#ifdef _WIN32
    mkdir(STR_LOGS);
#else
    mkdir(STR_LOGS, 0755);
#endif

    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
    params->mBcInt = 0;
}

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*)extmil1553ud_dmaBlock_getPtrDataPart(block->mData);
        unsigned short* dataWord16Etalon = (unsigned short*)
            ((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 КИИ
        unsigned short* dataWord16 =
            (unsigned short*)extmil1553ud_dmaBlock_getPtrDataPart(block->mData);
        for (unsigned int i = 0; i < 7; i++) {
            if (dataWord16[i] != CHESS_DATA) return MIL1553UD_FALSE;
        }
    }
}

```

Из	Под	Да

```

    }
    return MIL1553UD_TRUE;
}

int isRun(Abonent abonent)
{
    int isGood;
    SADDR_DATA regBcCondCode;
    unsigned int numChannel = abonent.extFd.channel;

    regBcCondCode.daddr = MIL_REG(numChannel, GPF_BC_COND_CODE_CH);
    isGood = IS_EXT_WRAPPER(abonent, mill1553_readReg, &regBcCondCode);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mill1553_readReg
GPF_BC_COND_CODE_CH\n", abonent.name);

    return (regBcCondCode.data & (1u << 31)) ? MIL1553UD_TRUE : MIL1553UD_FALSE;
}

char* stringsTransaction[11] = {"формат 0",
                                "формат 1 КШ -> ОУ (КС)",
                                "формат 2 ОУ -> КШ (КС)",
                                "формат 3 ОУ -> ОУ (КС)",
                                "формат 4 КУ",
                                "формат 5 КУ с СД (ОУ -> КШ)",
                                "формат 6 КУ с СД (КШ -> ОУ)",
                                "формат 7 КШ -> ОУ групповая (КС)",
                                "формат 8 ОУ -> ОУ групповая (КС)",
                                "формат 9 КУ групповая",
                                "формат 10 КУ с СД (КШ -> ОУ) групповая"};

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

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(unsigned char * block) {
    int index;
    unsigned int typeDmaBlock = MIL1553UD_TD_DMA_BLOCK; // ОУ
    unsigned int countServiceWords16 =
extmill1553ud_dmaBlock_getCountServiceWords16(typeDmaBlock, (unsigned char *)
block);
    unsigned short* block16 = (unsigned short*)block;
    int shift16 = countServiceWords16 - 1;
    unsigned int error = block16[shift16];
}

```

Из	Под	Да

```

LOGGER_PRINT("%s\n", "Сервисные слова блока ПДП(DMA) в сыром виде:");
for (index = 0; index < countServiceWords16; index++)
{
    LOGGER_PRINT("0x%x, ", block16[index]);
}

LOGGER_PRINT("%s", "\n");
LOGGER_PRINT("Значение регистра ошибок = 0x%x:\n", error);
for (index = 0; index < 16; index++) {
    if (error & (1 << index)) {
        LOGGER_PRINT("%s;\n", errorsBcString[index]);
    }
}
}

#define NEXT_BLOCKS_THRESHOLD 50

MIL1553_DMA_FIX_BLOCK create_dma_container()
{
    MIL1553_DMA_FIX_BLOCK container;

    memset(&container, 0, sizeof(MIL1553_DMA_FIX_BLOCK));
    container.mallocedBlocks = 256;
    container.data = (unsigned char*) malloc(MIL1553_ONE_BLOCK_SIZE_128_BYTES *
container.mallocedBlocks);

    return container;
}

void processingBc(void)
{
    int index = 0; // индекс КИИ
    MIL1553_DMA_FIX_BLOCK container = create_dma_container();

    if (configChannels.mIsReady[index])
    {
        if (MIL1553UD_FALSE == isBcStopped)
        {
            Abonent abonent = open_abonent_from_string(params.mNames[index]);
            int isRunValue = isRun(abonent);

            if (MIL1553UD_FALSE == isRunValue)
                startBc();
        }

        do
        {
            int ret, indexBlk;
            Abonent abonent = open_abonent_from_string(params.mNames[index]);

            uint64_t timeStart = getTimeStamp();

            ret = IS_EXT_WRAPPER(abonent, mil1553_readDmaFixBlocks, &container);
            // чтение dma блоков

            uint64_t timeEnd = getTimeStamp();
            printf("%s: processingBc - mil1553_readDma : %ld us\n", abonent.name,
timeEnd - timeStart);
            if (ret != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mil1553_readDma\n",
abonent.name);

```

Из	Под	Да

```

        if (GOOD_CODE == ret)
        {
            for (indexBlk = 0; indexBlk < container.currentBlocks; indexBlk++)
            {
                unsigned char* block = container.data + (indexBlk *
MIL1553_ONE_BLOCK_SIZE_128_BYTES);
                unsigned int typeDmaBlock = MIL1553UD_BC_DMA_BLOCK; // КШ
                unsigned int isSuccessTransaction =

extmil1553ud_dmaBlock_isSuccessTransaction(typeDmaBlock, block);
                unsigned int typeTransaction =
                    extmil1553ud_dmaBlock_getTypeTransaction(typeDmaBlock,
block);
                unsigned int freeTimer =
                    extmil1553ud_dmaBlock_getFreeTimer(typeDmaBlock,
block);
                unsigned int countServiceWords16 =

extmil1553ud_dmaBlock_getCountServiceWords16(typeDmaBlock, block);
                unsigned int countDataWords16 =

extmil1553ud_dmaBlock_getCountDataWords16(typeDmaBlock, block);
                unsigned int activeBus =
                    extmil1553ud_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 в КШ(BC): ", "Ошибка");
                    LOGGER_PRINT(
                        "Freetimer= %u, Сообщение= %d (%s), Шина= %s,
Сервисные слова= "
                        "%d, Слова данных= %d\n",
                        freeTimer, typeTransaction,
                        getMilTransactionString(typeTransaction),
                        (activeBus == MIL1553_BUS_A) ? "А" : "Б",
countServiceWords16,
                        countDataWords16);
                    printBcError(block);
                    if ((typeTransaction == 1) | (typeTransaction == 2))
                    {
                        if ((countDataWords16 > 0) && (isSuccessTransaction))
                        {
                            unsigned short* dataWord16 =
                                (unsigned
short*)extmil1553ud_dmaBlock_getPtrDataPart(block);
                            LOGGER_PRINT("%s: ", "Слова данных");
                            for (unsigned int i = 0; i < countDataWords16;
i++)
                                {
                                    if (i == countDataWords16 - 1) {
                                        LOGGER_PRINT("0x%x; \n", dataWord16[i]);

```

Из	Под	Да



```

        } else {
            LOGGER_PRINT("0x%x", dataWord16[i]);
        }
    }
}
}
}
}
}
if (TR_SYNCRO == typeTransaction)
{
    generateDataRt(abonent);
}
}
else
{
    LOGGER_PRINT("Ошибка ПДП(DMA) в КШ(BC) (%s) = %d\n",
        params.mNames[index], ret);
}
}
while(container.nextBlocks > NEXT_BLOCKS_THRESHOLD);
}

free(container.data);
}

void generateDataRt(Abonent abonent)
{
    int isGood;

    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++)
    {
        isGood = IS_EXT_WRAPPER(abonent, mil1553_rt_writeDataToSubaddressTransmit,
MIL_RT_BUF0, subaddress, data0);
        if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка
mil1553_rt_writeDataToSubaddressTransmit MIL_RT_BUF0\n", abonent.name);

        isGood = IS_EXT_WRAPPER(abonent, mil1553_rt_writeDataToSubaddressTransmit,
MIL_RT_BUF1, subaddress, data0);
        if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка
mil1553_rt_writeDataToSubaddressTransmit MIL_RT_BUF1\n", abonent.name);
    }
}

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.",

```

Из	Под	Да

```

"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(unsigned char * block) {
    int index;
    unsigned int typeDmaBlock = MIL1553UD_TD_DMA_BLOCK; // ОУ
    unsigned int countServiceWords16 =
        extmill1553ud_dmaBlock_getCountServiceWords16(typeDmaBlock, block);
    unsigned short* block16 = (unsigned short*)block;
    int shift16 = countServiceWords16 - 1;
    unsigned int error = block16[shift16];
    LOGGER_PRINT("%s\n", "Сервисные слова блока ПДП(DMA) в сыром виде:");
    for (index = 0; index < countServiceWords16; index++) {
        LOGGER_PRINT("0x%x, ", block16[index]);
    }
    LOGGER_PRINT("%s", "\n");
    LOGGER_PRINT("Значение регистра ошибок = 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;
    MIL1553_DMA_FIX_BLOCK container = create_dma_container();

    for (index = 1; index < COUNT_CH; index++)
    { // цикл по ОУ
        if (configChannels.mIsReady[index] == TRUE_VAL)
            do
            {
                int ret, indexBlk;

                Abonent abonent = open_abonent_from_string(params.mNames[index]);

                uint64_t timeStart = getTimeStamp();

                ret = IS_EXT_WRAPPER(abonent, mill1553_readDmaFixBlocks, &container);
// чтение dma блоков

                uint64_t timeEnd = getTimeStamp();
                printf("%s: processingRts [%d] - mill1553_readDma : %ld us\n",
abonent.name, index, timeEnd - timeStart);

                if(ret != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mill1553_readDma\n",
abonent.name);

                if (GOOD_CODE == ret)
                {
                    for (indexBlk = 0; indexBlk < container.currentBlocks; indexBlk++)
                    {
                        unsigned char* block = container.data + (indexBlk *
MIL1553_ONE_BLOCK_SIZE_128_BYTES);

```

Из	Под	Да

```

        unsigned int typeDmaBlock = MIL1553UD_TD_DMA_BLOCK; // ОУ
        unsigned int isSuccessTransaction =

extmil1553ud_dmaBlock_isSuccessTransaction(typeDmaBlock, block);
        unsigned int typeTransaction =
            extmil1553ud_dmaBlock_getTypeTransaction(typeDmaBlock,
block);

        unsigned int freeTimer =
            extmil1553ud_dmaBlock_getFreeTimer(typeDmaBlock,
block);

        unsigned int countServiceWords16 =
            extmil1553ud_dmaBlock_getCountServiceWords16(typeDmaBlock, block);
        unsigned int countDataWords16 =
            extmil1553ud_dmaBlock_getCountDataWords16(typeDmaBlock, block);
        unsigned int subaddress =
            extmil1553ud_dmaBlock_tdModeSubaddress(block);
        unsigned int activeBus =
            extmil1553ud_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(
                "Ошибка в ОУ(RT)-%d: Freetimer= %u, Сообщение=
%d (%s), Шина= "
                "%s, Подадрес= %d, Сервисные слова= %d, Слова
данных= %d\n",
                index, freeTimer, typeTransaction,
                getMilTransactionString(typeTransaction),
                (activeBus == MIL1553_BUS_A) ? "А" : "Б",
                countServiceWords16, countDataWords16);
            printRtError(block);
            if (typeTransaction == 1) {
                if ((countDataWords16 > 0) && (isSuccessTransaction))
                {
                    unsigned short* dataWord16 =
                        (unsigned
short*)extmil1553ud_dmaBlock_getPtrDataPart(block);
                    LOGGER_PRINT("%s: ", "Слова данных");
                    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("Ошибка ПДП(DMA) в ОУ(RT) (%s) = %d\n",
                params.mNames[index], ret);

```

Из	Под	Да

```

    }
    }
    while(container.nextBlocks > NEXT_BLOCKS_THRESHOLD);
  }

  free(container.data);
}

void readDevices(void)
{
#ifdef _WIN32
  devicesCount = 0;
#else
  int index;

  // построение массива обнаруженных девайсов
  mill1553_getArrayDevices(&devices, &devicesCount);
  for (index = 0; index < devicesCount; index++) {
    // Abonent abonent =
    open_abonent_from_string(devices[index].mDev.mNameChannel);
    devices[index].mDev.mFD = mill1553_open(devices[index].mDev.mNameChannel);
    VERSION ver;
    memset(&ver, 0, sizeof(VERSION));

    if(GOOD_CODE != mill1553_getDeviceInfo(devices[index].mDev.mFD, &ver))
    {
      LOGGER_PRINT(
        "Девайс: '%s' ОШИБКА mill1553_getDeviceInfo\n",
devices[index].mDev.mNameChannel);
    }

    unsigned int verValue = 0;
    if(GOOD_CODE != mill1553_getDriverVersion(devices[index].mDev.mFD,
&verValue))
    {
      LOGGER_PRINT(
        "Девайс: '%s' ОШИБКА mill1553_getDriverVersion\n",
devices[index].mDev.mNameChannel);
    }

    PCI_LOCATION loc;
    memset(&loc, 0, sizeof(PCI_LOCATION));
    if(GOOD_CODE != mill1553_getDevicePciInfo(devices[index].mDev.mFD, &loc))
    {
      LOGGER_PRINT(
        "Девайс: '%s' ОШИБКА mill1553_getDevicePciInfo\n",
devices[index].mDev.mNameChannel);
    }

    LOGGER_PRINT(
      "Девайс: '%s' deviceId= %d, revisionId= %d, pciloc= %s,
verValue= "
      "0x%x\n",
devices[index].mDev.mNameChannel, ver.device_id, ver.revision,
loc.name,
      verValue);

    mill1553_close(devices[index].mDev.mFD);
    devices[index].mDev.mFD = -1;
  }
#endif
}

```

Из	Под	Да

```

}

void readChannels(void) {
#ifdef _WIN32
    int index;
    // построение массива обнаруженных каналов
    mill1553_getArrayChannels(&channels, &channelsCount);
    for (index = 0; index < channelsCount; index++) {
        int numDev = mill1553_getNumberDevice(&(channels[index]));
        LOGGER_PRINT("Девайс= %d - Канал: '%s'\n", numDev,
                    channels[index].mNameChannel);
    }
#endif
}

void printResults(void) {
    PRINT_LINE();
    LOGGER_PRINT_STR("Тест завершён.\n");
    if (TRUE_VAL == configChannels.mIsReady[0])
        LOGGER_PRINT("bc %s: dmablк= %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: dmablк= %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: dmablк= %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: dmablк= %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: dmablк= %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: dmablк= %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: dmablк= %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: dmablк= %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: dmablк= %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])
        {
            Abonent abonent = open_abonent_from_string(params.mNames[index]);

            if ((!abonent.isExt && abonent.fd < 0) ||
                (abonent.isExt && abonent.extFd.socketFD < 0))
            {
                LOGGER_PRINT("Неправильное имя канала по индексу = %d, имя =
                '%s'\n",

```

Из	Под	Да

```

        index, params.mNames[index]);
    ret = FALSE_VAL;
}
}
}
return ret;
}

void bcDebugProcessing(void) {
    int index = 0; // индекс КИИ
    if (configChannels.mIsReady[index]) {
        SADDR_DATA regBcCondCode, regInstrPtr, regOpDataPtr;

        Abonent abonent = open_abonent_from_string(params.mNames[index]);
        unsigned int numChannel = abonent.extFd.channel;

        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);

        IS_EXT_WRAPPER(abonent, mil1553_readReg, &regBcCondCode);
        IS_EXT_WRAPPER(abonent, mil1553_readReg, &regInstrPtr);
        IS_EXT_WRAPPER(abonent, mil1553_readReg, &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) &
0xFFFF) * 4;
        IS_EXT_WRAPPER(abonent, mil1553_readReg, &instr);

        LOGGER_PRINT(
            " REG_INSTR_PTR: sinstr_ptr= 0x%x, cinstr_ptr= 0x%x,
instruction= "
            "0x%x\n",
            (regInstrPtr.data >> 16) & 0xFFFF, (regInstrPtr.data & 0xFFFF),
            (instr.data, 16));

        SADDR_DATA oper;
        oper.daddr = MIL_REG(numChannel, BC_OPERATION_RAM_CH) +
            ((regOpDataPtr.data >> 16) & 0xFFFF) * 4;
        IS_EXT_WRAPPER(abonent, mil1553_readReg, &oper);

        LOGGER_PRINT(
            " REG_OP_DATA_PTR: cop_ptr= 0x%x, cdat_ptr= 0x%x, operation=
0x%x\n",
            (regOpDataPtr.data >> 16) & 0xFFFF, (regOpDataPtr.data &
0xFFFF),
            (oper.data, 16));

        IS_EXT_WRAPPER(abonent, mil1553_close);
    }
}

```

Из	Под	Да

```

#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; \
    }

#define MAX_INTERRUPT_BLOCKS 64
INTERRUPT_BLOCK_BUFFER intBuffer;
struct block_info interruptBlocks[MAX_INTERRUPT_BLOCKS];

void testBcInt(void) {
    int countReadedBlocks = 0;
    int count = 0;
    int fd = 0;
    // LOGGER_PRINT("%s intBuffer ptr = 0x%x, intBuffer.blocks ptr = 0x%x \n",
    "КШ(BC) Чтение прерываний", &intBuffer, intBuffer.blocks);

    Abonent abonent = open_abonent_from_string(params.mNames[0]);
    do {
        count = IS_EXT_WRAPPER(abonent,
mill1553ud_getCountInterruptBlocksReadyRead);
        intBuffer.count_blocks = MAX_INTERRUPT_BLOCKS;
        unsigned int result = IS_EXT_WRAPPER(abonent,
mill1553ud_readInterruptBlocks, &intBuffer);
        if (result != GOOD) {
            LOGGER_PRINT("%s\n", "Ошибка функции mill1553ud_readInterruptBlocks");
        }
        countReadedBlocks += intBuffer.count_blocks;
    } while (count > 0);
    IS_EXT_WRAPPER(abonent, mill1553_close);

    // LOGGER_PRINT("КШ(BC) - зафиксировано прерываний = %d; intBuffer ptr = 0x%x,
intBuffer.blocks ptr = 0x%x %s", countReadedBlocks, &intBuffer, intBuffer.blocks,
"\n");
    LOGGER_PRINT("КШ(BC) - зафиксировано прерываний = %d %s", countReadedBlocks,
"\n");
}

void procCTest1(struct TParamCTest1 params) {
    int isValid;
    int countDevices;
    struct timeval tvPrev, tvCur, tvDiff;
    unsigned int allDurationSeconds;
    unsigned int curDurationSeconds;
    gettimeofday(&tvPrev, NULL);
    intBuffer.blocks = interruptBlocks;

#ifdef _WIN32
    countDevices = mill1553_getCountDevices();
    if (countDevices == 0) { // проверяем, что девайсов нет - значит и теста нет
        LOGGER_PRINT_STR(
            "Не найдено ни одного девайса. Невозможно провести тест.\n");
        return;
    }
#endif
    readDevices(); // печатаем список всех девайсов

```

Из	Под	Да

```

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("Неверное имя канала.\n");
    return;
}
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("Подождите пока завершится тест, пожалуйста...\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;
        if (params.mBcInt) {
            testBcInt();
        }
        //LOGGER_PRINT_STR("processingBc\n");
        processingBc(); // обработка блоков ДМА КШ
        //LOGGER_PRINT_STR("processingRts\n");
        processingRts(); // обработка блоков ДМА ОУ
        //LOGGER_PRINT_STR("STEP_SECONDS\n");
        STEP_SECONDS(1, curDurationSeconds, tvPrev, tvCur,
            tvDiff); // отсчитываем по одной секунде
        //LOGGER_PRINT_STR("end while\n");
    } while (curDurationSeconds <
        allDurationSeconds); // проверка завершения теста по длительности
    LOGGER_PRINT_STR("stopBc\n");
    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("Длительность теста = %d секунд\n", allDurationSeconds);

    return;
}

```

Из	Под	Да



## ПРИЛОЖЕНИЕ В (ИНФОРМАЦИОННОЕ)

## Пример визуализации тестовой программы

```

parsing './config/test.txt':
duration_h 8          //продолжительность теста часы
duration_m 5          //продолжительность теста минуты
line 0                //линия: 0-A; 1-B
bc mill1553dev-0-ch-1 //КШ
rt1 mill1553dev-0-ch-0 //OY1
rt2 mill1553dev-0-ch-2 //OY2
rt3 mill1553dev-0-ch-3 //OY3
rt4 mill1553dev-usb-0-ch-0 //OY4
rt5 192.168.10.10:19191:0 //OY5
rt6 192.168.10.10:19191:1 //OY6
rt7 NULL              //OY7
rt8 NULL              //OY8

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-usb-0-ch-0,
rt5= /dev/192.168.10.10:19191:0,
rt6= /dev/192.168.10.10:19191:1,
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-usb-0-ch-0' deviceId= 38002, revisionId= 3
Device: '/dev/192.168.10.10:19191:0' deviceId= 38002, revisionId= 3
Device: '/dev/192.168.10.10:19191:1' 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!

```

Из	Под	Да

```
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: dmablк= 142316289; err= 0;  
rt1 /dev/mil1553dev-0-ch-0: dmablк= 29961324; err= 0;  
rt2 /dev/mil1553dev-0-ch-2: dmablк= 29961324; err= 0;  
rt3 /dev/mil1553dev-0-ch-3: dmablк= 29961324; err= 0;  
rt4 /dev/mil1553dev-usb-0-ch-0: dmablк= 29961324; err= 0;  
rt5 /dev/192.168.10.10:19191:0: dmablк= 29961324; err= 0;  
rt6 /dev/192.168.10.10:19191:1: dmablк= 29961324; err= 0;  
rt7 NULL: dmablк= 0; err= 0;  
rt8 NULL: dmablк= 0; err= 0;  
-----  
Time spent on the test = 29100 seconds
```

Рисунок В1 - Пример визуализации тестовой программы

Из	Под	Да

## 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);

Из	Под	Да

