УТВЕРЖДАЮ

Генеральный директор

ООО «НОВОМАР»

_____ Т.В. Буга

«____»_____2022 г.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ «БИБЛИОТЕКА СЕТЕВОГО ВЗАИМОДЕЙСТВИЯ MIL1553UD»

Модулей
“LAN–MIL1553UDx”

(ОС WINDOWS, ОС LINUX)

Руководство программиста

ЛИСТ УТВЕРЖДЕНИЯ

RU.МСКЮ.20106-01 33 01-ЛУ

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

От

_____

«____»_____2022 г.

Инженер-программист

_____ _____

«____»_____2022 г.

2022

Из	Под	Да

Литера

Утвержден

RU.МСКЮ.20106-01 33 01-ЛУ

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«БИБЛИОТЕКА СЕТЕВОГО ВЗАИМОДЕЙСТВИЯ MIL1553UD»

Модулей
“LAN–MIL1553UDx”

(ОС WINDOWS, ОС LINUX)
Руководство программиста

RU.МСКЮ.20106-01 33 01

Листов 55

2022

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

Из	Под	Да

Литера

АННОТАЦИЯ

В книге описываются технологические принципы, использованные в программном обеспечении «БИБЛИОТЕКА СЕТЕВОГО ВЗАИМОДЕЙСТВИЯ MIL1553UD» (ОС Linux и Astra Linux, а также Windows 10) версий 3.x, для работы модулей “LAN–MIL1553UDx” в сети МКИО ГОСТ Р 52070-2003. В частности, рассмотрены функциональное назначение и область применения, условия выполнения.

СОДЕРЖАНИЕ

1 НАЗНАЧЕНИЕ ПРОГРАММЫ

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

Библиотека обеспечивает выполнение следующих основных задач:

– реализация сервисных функций с использованием механизма сетевых сокетов, для работы с LAN–MIL1553UDx

2 УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

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

3 ХАРАКТЕРИСТИКА ПРОГРАММЫ

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

- mil1553ud_cmd.h
- mil1553ud_common.h
- mil1553ud_regs.h
- mil1553udext_eth_ioctl.h
- mil1553udext_ioctl_wrappers.h
- mil1553udext_library.h
- mil1553udext_meu_proto.h
- mil1553udext_eth_ioctl.c
- mil1553udext_library.c

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

В случае ОС Windows, также необходимо обеспечить линковку проекта с библиотекой WebSocket2. Например, в случае системы сборки qmake, добавьте такую строку в pro-файл проекта:

```
win32: LIBS += -lws2_32
```

4 ОБРАЩЕНИЕ К ПРОГРАММЕ

4.1 ОБЩИЕ СВЕДЕНИЯ

Библиотека сетевого взаимодействия предназначена для функционирования в ОС Linux, Astra Linux и ОС Windows. Её возможно использовать как в статическом, динамическом виде, так и с помощью встраивания в прикладное ПО как исходный код.

Обмен данных с платой LAN–MIL1553UDx ведётся с использованием API сетевых сокетов. Взаимодействие с каналами происходит посредством функций, представленных в заголовочном файле «mil1553udext_library.h».

В случае API сетевых сокетов, сервисные функции в качестве первого аргумента принимают структуру-дескриптор сетевого канала (`Mil1553ChannelFD cfd`). открыть который необходимо при помощи `extmil1553_open`, а для корректного завершения работы — закрыть методом `extmil1553_close`.

Пример исходного кода программы-примера приведён в приложении А.

Скриншоты работы тестовой программы приведены в приложении Б.

4.2 МЕТОДЫ БИБЛИОТЕКИ СЕТЕВОГО ВЗАИМОДЕЙСТВИЯ

4.2.1 Генерация имени регистра в соответствии с номером канала

Описание функции в файле «mil1553udext_library.h» см. на рисунке 1.

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

Рисунок 1 – Листинг функции

4.2.2 Получить количество блоков DMA, готовых для чтения

Описание функции в файле «mil1553udext_library.h» см. на рисунке 2.

```
///
/// \brief получить количество блоков ДМА, готовых для чтения
/// \param cfd дескриптор канала
/// \param codeError код возврата (признак ошибки)
/// \return кол-во блоков ДМА, готовых к чтению
///
unsigned int extmil1553ud_getCountDmaBlocksReadyRead(Mil1553ChannelFD cfd,
unsigned int* codeError);
```

Рисунок 2 – Листинг функции

4.2.3 Создать контейнер блоков DMA

Описание функции в файле «mil1553udext_library.h» см. на рисунке 3.

```
///
/// \brief создать контейнер блоков ДМА
/// выделяет под контейнер память
/// \param countDmaBlocks размер контейнера в блоках (штуках)
/// \return указатель на контейнер
///
DMA_READ_BLOCK* extmil1553ud_newDmaReadBlock(unsigned int countDmaBlocks);
```

Рисунок 3 – Листинг функции

4.2.4 Освобождает контейнер блоков DMA

Описание функции в файле «mil1553udext_library.h» см. на рисунке 4.

```
/// \brief освобождает контейнер блоков ДМА
/// \param block указатель на контейнер
///
void extmil1553ud_freeDmaReadBlock(DMA_READ_BLOCK* block);
```

Рисунок 4 – Листинг функции

4.2.5 Считывает все блоки DMA, готовые к чтению, из канала

Описание функции в файле «mil1553udext_library.h» см. на рисунке 5.

```
/// \brief считывает все блоки ДМА, готовые к чтению, из канала
/// выделяет под контейнер память
/// \param cfd дескриптор канала
/// \return указатель на контейнер
///
DMA_READ_BLOCK* extmil1553ud_readAllDmaBlocksReadyRead(Mil1553ChannelFD
cfd, unsigned int *cntBlocks);
```

Рисунок 5 – Листинг функции

4.2.6 Получить количество блоков прерываний, готовых для чтения

Описание функции в файле «mil1553udext_library.h» см. на рисунке 6.

```
/// \brief получить количество блоков прерываний, готовых для чтения
/// \param cfd дескриптор канала
/// \return кол-во блоков, готовых к чтению
///
unsigned int extmil1553ud_getCountInterruptBlocksReadyRead(Mil1553ChannelFD
cfd);
```

Рисунок 6 – Листинг функции

4.2.7 Считывает блоки прерываний, готовые к чтению, из канала

Описание функции в файле «mil1553udext_library.h» см. на рисунке 7.

```
/// \brief считывает блоки прерываний, готовые к чтению, из канала
/// \param cfd дескриптор канала
/// \param block указатель на контейнер
/// \return результат операции
///
unsigned int extmil1553ud_readInterruptBlocks(Mil1553ChannelFD cfd,
INTERRUPT_BLOCK_BUFFER* block);
```

Рисунок 7 – Листинг функции

4.2.8 Управление микропрограммой в режиме КШ

Описание функции в файле «mil1553udext_library.h» см. на рисунке 8.

```
///
/// \brief управление микропрограммой в режиме КШ
/// старт/стоп BCSTRT
/// \param cfd дескриптор канала
/// \param action действие (BC_PROGRAM_START, BC_PROGRAM_STOP)
///
unsigned int extmil1553ud_bcProgram(Mil1553ChannelFD cfd, unsigned int
action);

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

Рисунок 8 – Листинг функции

4.2.9 Установить стартовый адрес инструкции микропрограммы КШ

Описание функции в файле «mil1553udext_library.h» см. на рисунке 9.

```
///
/// \brief установить стартовый адрес интсрукции микропрограммы КШ
/// \param cfd дескриптор канала
/// \param startNumber стартовый номер (адрес)
///
unsigned int extmil1553ud_bcSetStartInstructionAddress(Mil1553ChannelFD
cfd, unsigned int startNumber);
```

Рисунок 9 – Листинг функции

4.2.10 Создать контейнер для микропрограммы КШ

Описание функции в файле «mil1553udext_library.h» см. на рисунке 10.

```
///
/// \brief создать контейнер для микропрограммы КШ
/// \param type тип - инструкции, операции, данные
/// MIL_BC_RAM_TYPE_INSTRUCTION, MIL_BC_RAM_TYPE_OPERATION,
MIL_BC_RAM_TYPE_DATA
/// \param sizeInDwords размер контейнера в 32-разядных (двойных) словах
/// \param shiftInDwords смещение в памяти КШ в 32-разядных (двойных)
словах
/// \return указатель на контейнер для микропрограммы КШ
///
BC_RAM_BLOCK* extmil1553ud_newBcRamBlock(unsigned int type, unsigned int
sizeInDwords, unsigned int shiftInDwords);
```

Рисунок 10 – Листинг функции

4.2.11 Освободить контейнер для микропрограммы КШ

Описание функции в файле «mil1553udext_library.h» см. на рисунке 11.

```
///
/// \brief освободить контейнер для микропрограммы КШ
/// \param block указатель на контейнер для микропрограммы КШ
///
void extmil1553ud_freeBcRamBlock(BC_RAM_BLOCK* block);
```

Рисунок 11 – Листинг функции

4.2.12 Формирование инструкции для микропрограммы КШ

Описание функции в файле «mil1553udext_library.h» см. на рисунке 12.

```
///
/// \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 12 – Листинг функции

4.2.13 Формирование нечётной части операции для микропрограммы КШ

Описание функции в файле «mil1553udext_library.h» см. на рисунке 13.

```
///
/// \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 13 – Листинг функции

4.2.14Формирование чётной части операции для микропрограммы КШ

Описание функции в файле «mil1553udext_library.h» см. на рисунке 14.

```
///
/// \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 14 – Листинг функции

4.2.15Формирование командного слова операции для микропрограммы КШ

Описание функции в файле «mil1553udext_library.h» см. на рисунке 15.

```
///
/// \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 15 – Листинг функции

4.2.16Получить указатель на блок DMA из контейнера

Описание функции в файле «mil1553udext_library.h» см. на рисунке 16.

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

Рисунок 16 – Листинг функции

4.2.17Проверка на допустимость контейнера DMA

Описание функции в файле «mil1553udext_library.h» см. на рисунке 17.

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

Рисунок 17 – Листинг функции

4.2.18Получить тип транзакции из блока DMA

Описание функции в файле «mil1553udext_library.h» см. на рисунке 18.

```
///
/// \brief получить тип транзакции из блока ДМА
/// тип транзакции соттветствует значениям 1-10 (см. ГОСТ 52070)
/// \param dmaBlock тип блока ДМА
/// \param dmaBlock указатель на блок ДМА
/// \return тип транзакции
///
unsigned int extmil1553ud_dmaBlock_getTypeTransaction(unsigned int
typeDmaBlock, unsigned char* dmaBlock);
```

Рисунок 18 – Листинг функции

4.2.19Проверить успешность транзакции

Описание функции в файле «mil1553udext_library.h» см. на рисунке 19.

```
///
/// \brief получить значение freetimer блока ДМА
/// \param typeDmaBlock тип блока ДМА
/// \param dmaBlock указатель на блок ДМА
/// \return значение freetimer
///
unsigned int extmil1553ud_dmaBlock_getFreeTimer(unsigned int typeDmaBlock,
unsigned char* dmaBlock);
```

Рисунок 19 – Листинг функции

4.2.20Получить количество служебных 16-разрядных слов

Описание функции в файле «mil1553udext_library.h» см. на рисунке 20.

```
///
/// \brief 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 extmil1553ud_dmaBlock_getCountServiceWords16(unsigned
int typeDmaBlock, unsigned char* dmaBlock);
```

Рисунок 20 – Листинг функции

4.2.21Получить количество 16-разрядных слов данных

Описание функции в файле «mil1553udext_library.h» см. на рисунке 21.

```
///
/// \brief 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 extmil1553ud_dmaBlock_getCountDataWords16(unsigned int
typeDmaBlock, unsigned char* dmaBlock);
```

Рисунок 21 – Листинг функции

4.2.22Получить активную шину из блока DMA

Описание функции в файле «mil1553udext_library.h» см. на рисунке 22.

```
///
/// \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);
```

Рисунок 22 – Листинг функции

4.2.23Получить указатель на часть с данными блока DMA

Описание функции в файле «mil1553udext_library.h» см. на рисунке 23.

```
///
/// \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 23 – Листинг функции

4.2.24Получить подадрес из блока DMA в режиме ОУ

Описание функции в файле «mil1553udext_library.h» см. на рисунке 24.

```
///
/// \brief mil1553ud_dmaBlock_tdModeSubaddress получить поддарес из
блока DMA в режиме ОУ
/// \param dmaBlock указатель на блок DMA
/// \return поддарес
///
unsigned int extmil1553ud_dmaBlock_tdModeSubaddress(unsigned char*
dmaBlock);
```

Рисунок 24 – Листинг функции

4.2.25Открыть канал (внешнее сетевое устройство) на чтение и запись

Описание функции в файле «mil1553udext_library.h» см. на рисунке 25.

```
/// \brief подключиться к каналу связи
/// \param ipv4_addr адрес устройства (строка, например "192.168.10.10")
/// \param port порт устройства
/// \param channel mil-канал устройства
/// \return дескриптор открытого канала (проверьте, что socketFd не -1)
Mil1553ChannelFD extmil1553_open(const char *address, uint16_t port,
uint16_t channel);
```

Рисунок 25 – Листинг функции

4.2.26Закрыть канал (внешнее сетевое устройство)

Описание функции «mil1553udext_library.h» см. на рисунке 26.

```
/// \brief закрыть канал связи
/// \param cfd дескриптор канала
int extmil1553_close(Mil1553ChannelFD cfd);
```

Рисунок 26 – Листинг функции

4.2.27Чтение 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 «mil1553udext_library.h» см. на рисунке 27.

```
#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 чтение фиксированное кол-во блоков
/// \param cfd дескриптор канала
/// \param container контейнер блоков ДМА
/// \return результат операции
///
unsigned int extmil1553_readDmaFixBlocks(Mil1553ChannelFD cfd,
MIL1553_DMA_FIX_BLOCK* container);
```

Рисунок 27 – Листинг функции

4.2.28 Чтение блоков из ДМА буфера канала (устаревшая версия)

Описание функции в файле «mil1553udext_library.h» см. на рисунке 28.

```
/// \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 Чтение данных (блоков) из ДМА буфера канала
/// \param cfd дескриптор канала
/// \param container контейнер блоков ДМА
/// \return результат операции
///
unsigned int extmil1553_readDma(Mil1553ChannelFD cfd, struct
MIL1553_DMA_CONTAINER_BLOCKS* container);
```

Рисунок 28 – Листинг функции

4.2.29 Включение/выключение канала

Описание функции в файле «mil1553udext_library.h» см. на рисунке 29.

```
/// \brief Вкл./выкл. канала
/// \param cfd дескриптор канала
/// \param op MIL_DEV_CHANNEL_ON, MIL_DEV_CHANNEL_OFF
/// \return результат операции
///
unsigned int extmil1553_manChannel(Mil1553ChannelFD cfd, unsigned int op);
```

Рисунок 29 – Листинг функции

4.2.30 Выбор роли (режима) канала

Описание функции в файле «mil1553udext_library.h» см. на рисунке 30.

```
/// \brief Выбор роли (режима) канала КШ, ОУ, МШ, ОУМШ
/// \param cfd дескриптор канала
/// \param role роль (режим)
/// \return результат операции
///
unsigned int extmil1553_setRole(Mil1553ChannelFD cfd, unsigned int role);
```

Рисунок 30 – Листинг функции

4.2.31 Получить количество каналов заданного девайса (API внешнего устройства)

Описание функции в файле «mil1553udext_library.h» см. на рисунке 31.

```
///
/// \brief получить кол-во каналов внешнего устройства
/// \param cfd дескриптор канала
/// \return кол-во каналов
///
int extmil1553_getCountChannels(Mil1553ChannelFD cfd);
```

Рисунок 31 – Листинг функции

4.2.32 Записать значение в регистр

Описание функции в файле «mil1553udext_library.h» см. на рисунке 32.

```
/// \brief записать значение регистра
/// \param cfd дескриптор канала
/// \param reg инфо о регистре
/// \return результат операции
///
unsigned int extmil1553_writeReg(Mil1553ChannelFD cfd, SADDR_DATA* reg);
```

Рисунок 32 – Листинг функции

4.2.33 Прочитать значение из регистра

Описание функции в файле «mil1553udext_library.h» см. на рисунке 33.

```
/// \brief прочитать значение регистра
/// \param cfd дескриптор канала
/// \param reg инфо о регистре
/// \return результат операции
///
unsigned int extmil1553_readReg(Mil1553ChannelFD cfd, SADDR_DATA* reg);
```

Рисунок 33 – Листинг функции

4.2.34Изменить заданные биты регистра

Описание функции в файле «mil1553udext_library.h» см. на рисунке 34.

```
/// \brief изменить заданные биты регистра
/// \param cfd дескриптор канала
/// \param reg инфо о регистре
/// \return результат операции
///
unsigned int extmil1553_modifyReg(Mil1553ChannelFD cfd,
SADDR_DATA_BIT_MASK* reg);
```

Рисунок 34 – Листинг функции

4.2.35Получить информацию о канале и плате

Описание функции в файле «mil1553udext_library.h» см. на рисунке 35.

```
/// \brief получить информацию о канале и плате
/// \param cfd дескриптор канала
/// \param info инфо о канале и плате
/// \return результат операции
///
unsigned int extmil1553_getDeviceInfo(Mil1553ChannelFD cfd, VERSION* info);
```

Рисунок 35 – Листинг функции

4.2.36Получить версию драйвера

Описание функции в файле «mil1553udext_library.h» см. на рисунке 36.

```
/// \brief получить версию драйверу
/// \param cfd дескриптор канала
/// \param version версия драйвера
/// \return результат операции
///
unsigned int extmil1553_getDriverVersion(Mil1553ChannelFD cfd, unsigned
int* version);
```

Рисунок 36 – Листинг функции

4.2.37Сброс ДМА

Описание функции в файле «mil1553udext_library.h» см. на рисунке 37.

```
///
/// \brief Сброс ДМА
/// \param cfd дескриптор канала
/// \return результат операции
///
unsigned int extmil1553_clearDma(Mil1553ChannelFD cfd);
```

Рисунок 37 – Листинг функции

4.2.38 Управление ДМА

Описание функции в файле «mil1553udext_library.h» см. на рисунке 38.

```
///
/// \brief управление ДМА
/// \param cfd дескриптор канала
/// \param op 0 - выкл, !0 - вкл
/// \return результат операции
///
unsigned int extmil1553_manDma(Mil1553ChannelFD cfd, int op);
```

Рисунок 38 – Листинг функции

4.2.39 Выбор шины А и/или Б

Описание функции в файле «mil1553udext_library.h» см. на рисунке 39.

```
///
/// \brief Выбор шины А и/или Б
/// \param cfd дескриптор канала
/// \param bus шина - MIL_BUS_A_EN, MIL_BUS_B_EN
/// \return результат операции
///
unsigned int extmil1553_switchBus(Mil1553ChannelFD cfd, unsigned int bus);
```

Рисунок 39 – Листинг функции

4.2.40Запись данных в RAM КШ

Описание функции в файле «mil1553udext_library.h» см. на рисунке 40.

```
///
/// \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 запись данных в RAM контроллера шины
/// \param cfd дескриптор канала
/// \param type тип озу контроллера шины
/// \param data данные (массив)
/// \param length размер массива
/// \return результат операции
///
unsigned int extmil1553_bc_writeDataToRam(Mil1553ChannelFD cfd, enum
EMil1553BcTypeDataRam type, unsigned int* data, int length);
```

Рисунок 40 – Листинг функции

4.2.41 Разрешение/запрет подадресов ОУ на приём/передачу

Описание функции в файле «mil1553udext_library.h» см. на рисунке 41.

```
///
/// \brief Разрешение подадресов ОУ на приём/передачу
/// \param cfd дескриптор канала
/// \param direction направление MIL_RT_BUF_RECEIVE, MIL_RT_BUF_TRANSMIT
/// \param subadr_mask маска подадресов (1 - изменять подадрес, 0 - не
изменять поадрес) (с 1 по 30 подадрес)
/// \param action_mask маска действий (1 - разрешить, 0 - запретить) (с 1
по 30 подадрес)
/// \return результат операции
///
unsigned int extmil1553_rt_manSubaddresses(Mil1553ChannelFD cfd, unsigned
int direction, unsigned int subadr_mask, unsigned int action_mask);
```

Рисунок 41 – Листинг функции

4.2.42 Установить адрес ОУ

Описание функции в файле «mil1553udext_library.h» см. на рисунке 42.

```
///
/// \brief установить адрес ОУ
/// \param cfd дескриптор канала
/// \param address адрес ОУ
/// \return результат операции
///
unsigned int extmil1553_rt_setAddress(Mil1553ChannelFD cfd, unsigned char
address);
```

Рисунок 42 – Листинг функции

4.2.43 Управление режимом буфера ОУ на передачу

Описание функции в файле «mil1553udext_library.h» см. на рисунке 43.

```
///
/// \brief управление режимом буфера ОУ на передачу
/// \param cfd дескриптор канала
/// \param subaddress подадрес
/// \param mode режим буфера MIL_RT_MODE_PROG, MIL_RT_MODE_HRDW
/// \return результат операции
///
unsigned int extmil1553_rt_manSubadressTransmitBufferMode(Mil1553ChannelFD
cfd, unsigned char subaddress, unsigned int mode);
```

Рисунок 43 – Листинг функции

4.2.44Управление готовностью буфера ОУ на передачу

Описание функции в файле «mil1553udext_library.h» см. на рисунке 44.

```
///
/// \brief управление готовностью буфера ОУ на передачу
/// \param cfd дескриптор канала
/// \param subaddress подадрес
/// \param cmd_buf команды управления буферами
/// \return результат операции
///
unsigned int
extmil1553_rt_manSubaddressTransmitBufferReady(Mil1553ChannelFD cfd,
unsigned char subaddress, unsigned int cmd_buf);
```

Рисунок 44 – Листинг функции

4.2.45Запись данных в подадрес отправного буфера ОУ

Описание функции в файле «mil1553udext_library.h» см. на рисунке 45.

```
/// \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 запись данных в подадрес отправного буфера ОУ
/// \param cfd 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 extmil1553_rt_writeDataToSubaddressTransmit(Mil1553ChannelFD
cfd, unsigned int num_buf, unsigned char subaddress, struct
TRtSubaddressData data);
```

Рисунок 45 – Листинг функции

5 СООБЩЕНИЯ

Описание кодов ошибок приведено на рисунке 46.

```
/// \brief невалидное значение, свидетельствующее об ошибке
#define LIB_BAD_VALUE       0xFFFFFFFF
/// \brief значение отсутсвия ошибки
#define LIB_GOOD            0
/// \brief ошибка адреса регистра
#define ERR_BAD_ADDRESS                 -1001
/// \brief ошибка номера канала
/// внутрення ошибка в драйвере
#define ERR_BAD_CHANNEL                 -1002
/// \brief ошибка копирования данных в юзерспейс
#define ERR_COPY_TO_USER                -1003
/// \brief ошибка аргумента - mode
#define ERR_BAD_ARG_MODE                -1004
/// \brief ошибка аргумента - subaddress
#define ERR_BAD_ARG_SA                  -1005
/// \brief ошибка аргумента - direction
#define ERR_BAD_ARG_DIRECTION           -1006
/// \brief ошибка аргумента - tck
#define ERR_BAD_ARG_TCK                 -1007
/// \brief ошибка аргумента - trck
#define ERR_BAD_ARG_TRCK                -1008
/// \brief ошибка аргумента - rcvck
#define ERR_BAD_ARG_RCVCK               -1009
/// \brief ошибка аргумента - numbuf
#define ERR_BAD_ARG_NUMBUF              -1010
/// \brief ошибка аргумента - typr
#define ERR_BAD_ARG_TYPE                -1011
/// \brief ошибка выделения памяти
/// внутренняя ошибка драйвера
#define ERR_BAD_ALLOC_MEM               -1012
```

– Рисунок 46 - Листинг

ПРИЛОЖЕНИЕ А (ИНФОРМАЦИОННОЕ)

Листинг программы-примера

Листинг А1 – 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, учавствующих в тесте
#define COUNT_CH (8+1)
/// \brief шахматка
#define CHESS_DATA          0xAAAA
/// \brief обратная шахматка
#define REVERSE_CHESS_DATA  0x5555
/// \brief индекс канала КШ в массиве параметров
#define BC_CH_INDEX         0

///
/// \brief The SConfigCh struct каналы, готовые к работе
/// 0 - КШ, 1 - ОУ1 ...
///
struct SConfigCh {
    int mIsReady[COUNT_CH];     ///< признак готовности канала
    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
```

```
///
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-управляемый 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;         ///< конфигкрация готовности
каналов
extern struct TStatCTest1 statTest;             ///< статистика теста

///
/// \brief main точка входа в программу
/// \param argc кол-во аргументов
/// \param argv аргументы
/// \return код возврата
///
int main(int argc, char* argv[]);
```

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

#endif // _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 TMil1553Channel* channels;
int channelsCount;
struct TMil1553Device* devices;
int devicesCount;
struct SConfigCh configChannels;  ///< конфигкрация готовности каналов
struct TStatCTest1 statTest;  ///< статистика теста

static unsigned int isBcStopped;

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

```
typedef struct
{
    char name[ CHANNEL_NAME_LENGTH ];
    bool isExt;
    int fd;
    Mil1553ChannelFD 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)
            extmil1553_close(g_abonentCache->abonent.extFd);
#ifndef _WIN32
        else
            mil1553_close(g_abonentCache->abonent.fd);
#endif

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

```
        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 == strncmp(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 = extmil1553_open(str_ipAddr.c_str(), port, channel);
    }
    else
    {
#ifdef _WIN32
        abort();
#else
        // локальное устройство
```

```
        result.fd = mil1553_open(name);
        result.extFd.channel = mil1553ud_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
```

```
                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 КШ
static unsigned int memDesc[MEM_SIZE];
/// \brief операции КШ
static unsigned int memOp[MEM_SIZE];
/// \brief данные КШ
static unsigned int memData[MEM_SIZE];

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

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

    if ((countOfOnes % 2) == 0)
        *word = *word | (1u << 31);
    else
        *word = *word & ~(1u << 31);
}

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

```
void applyParametresToMem(void)
{
    int counter;
    if (0 == params.mLine)
    {  // off line b
        memDescRtOff(&memDesc[0x6]);
        memDescRtOff(&memDesc[0x7]);
        memDescRtOff(&memDesc[0x8]);
        memDescRtOff(&memDesc[0x9]);
    }
    else
    {  // off line a
        memDescRtOff(&memDesc[0x2]);
        memDescRtOff(&memDesc[0x3]);
        memDescRtOff(&memDesc[0x4]);
        memDescRtOff(&memDesc[0x5]);
    }

    for (counter = 0; counter < MAX_DESC_SHIFT_COUNTER; counter++)
    {
        // применение параметров к мем descriptors
        memDescRtOffWithCheckReady(1, counter);
        memDescRtOffWithCheckReady(2, counter);
        memDescRtOffWithCheckReady(3, counter);
        memDescRtOffWithCheckReady(4, counter);
        memDescRtOffWithCheckReady(5, counter);
        memDescRtOffWithCheckReady(6, counter);
        memDescRtOffWithCheckReady(7, counter);
        memDescRtOffWithCheckReady(8, counter);
    }
}

void writeMemToBc(void)
{
    if (configChannels.mIsReady[0])
    {
        int isGood;
        Abonent abonent = open_abonent_from_string(params.mNames[0]);

        isGood = IS_EXT_WRAPPER(abonent, mil1553_bc_writeDataToRam,
Mil1553BcTypeInstructions, 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, mil1553_bc_writeDataToRam,
Mil1553BcTypeOperations, 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, mil1553_bc_writeDataToRam,
Mil1553BcTypeData, memData, MEM_SIZE);
        LOGGER_PRINT_STR("./mem/consol_dat_ram.mem");
```

```
        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, mil1553_manChannel, opCh);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mil1553_manChannel\n",
abonent.name);

    unsigned int opDma = MIL1553_OFF;
    isGood = IS_EXT_WRAPPER(abonent, mil1553_manDma, opDma);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mil1553_manDma\n",
abonent.name);

    isGood = IS_EXT_WRAPPER(abonent, mil1553_clearDma);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mil1553_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, mil1553_setRole, modeBus);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mil1553_manChannel\n",
abonent.name);

    isGood = IS_EXT_WRAPPER(abonent, mil1553_switchBus, bus);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mil1553_manChannel\n",
abonent.name);

    if (0 != index)
    {
        isGood = IS_EXT_WRAPPER(abonent, mil1553_rt_setAddress, index);
        if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mil1553_manChannel\n",
abonent.name);
    }
}

void initChannel_initRtSubaddresses(Abonent abonent)
{
    int isGood;

    isGood = IS_EXT_WRAPPER(abonent, mil1553_rt_manSubaddresses,
                            MIL_RT_BUF_RECEIVE, 0x7FFFFFFE /* меняем с 1 по 30
подадрес*/,
                            0x7FFFFFFE /* включаем на приём с 1 по 30 подадрес*/);
```

```
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mil1553_rt_manSubaddresses
MIL_RT_BUF_RECEIVE\n", abonent.name);

    isGood = IS_EXT_WRAPPER(abonent, mil1553_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 mil1553_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,
mil1553_rt_manSubadressTransmitBufferMode, subaddress, MIL_RT_MODE_PROG);
        if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка
mil1553_rt_manSubadressTransmitBufferMode\n", abonent.name);

        isGood = IS_EXT_WRAPPER(abonent,
mil1553_rt_manSubaddressTransmitBufferReady, subaddress, MIL_RT_BUF_0_ON_1_OFF);
        if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка
mil1553_rt_manSubaddressTransmitBufferReady\n", abonent.name);
    }

    generateDataRt(abonent);
}

void initChannel_switchOnChannel(Abonent abonent)
{
    int isGood;

    // включаем ДМА
    unsigned int opDma = MIL1553_ON;
    isGood = IS_EXT_WRAPPER(abonent, mil1553_manDma, opDma);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mil1553_manDma MIL1553_ON\n",
abonent.name);

    // включаем канал
    unsigned int opCh = MIL_DEV_CHANNEL_ON;
    isGood = IS_EXT_WRAPPER(abonent, mil1553_manChannel, opCh);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mil1553_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)
            {
```

```
                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
подадресов
                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, mil1553_manChannel, opCh);
            if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mil1553_manChannel
MIL_DEV_CHANNEL_OFF\n", abonent.name);

            unsigned int opDma = MIL1553_OFF;
            isGood = IS_EXT_WRAPPER(abonent, mil1553_manDma, opDma);
            if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mil1553_manDma
MIL1553_OFF\n", abonent.name);

            isGood = IS_EXT_WRAPPER(abonent, mil1553_clearDma);
            if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mil1553_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]);
```

```
        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, mil1553_writeReg, &reg);
            if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mil1553_writeReg\n",
abonent.name);
        }

        isGood = IS_EXT_WRAPPER(abonent, mil1553ud_bcSetStartInstructionAddress,
BC_START_ADDRESS);
        if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка
mil1553ud_bcSetStartInstructionAddress\n", abonent.name);

        reg.daddr = MIL_REG(numCh, START_ADR_INSTR_PTR_CH);     // адрес регистра
        isGood = IS_EXT_WRAPPER(abonent, mil1553_readReg, &reg);
        if(isGood != GOOD_CODE)
            LOGGER_PRINT("%s: Ошибка mil1553_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, mil1553ud_bcProgram, BC_PROGRAM_START);
// BCSTRT - старт выполнения программы КШ;
        if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mil1553ud_bcProgram
BC_PROGRAM_START\n", abonent.name);

        reg.daddr = MIL_REG(numCh, BC_CONF_REG_PCI_CH);     // адрес регистра
        isGood = IS_EXT_WRAPPER(abonent, mil1553_readReg, &reg);
        if(isGood != GOOD_CODE)
            LOGGER_PRINT("%s: Ошибка mil1553_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);
    }
}
```

```
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, mil1553ud_bcProgram, BC_PROGRAM_STOP);
// BCSTRT - старт выполнения программы КШ;
        if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mil1553ud_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,
```

```
                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;
                        snprintf(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) {
```

```
    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 КШ
        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;
        }
```

```
    }
    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, mil1553_readReg, &regBcCondCode);
    if(isGood != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mil1553_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 =
extmil1553ud_dmaBlock_getCountServiceWords16(typeDmaBlock, (unsigned char *)
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", 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;  // инрдекс КШ
    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);
```

```
            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(ВС): ", "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) ? "A" : "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]);
```

```
                                    } else {
                                        LOGGER_PRINT("0x%x,", dataWord16[i]);
                                    }
                                }
                            }
                        }
                    }
                    if (TR_SYNCRO == typeTransaction)
                    {
                        generateDataRt(abonent);
                    }
                }
            }
            else
            {
                LOGGER_PRINT("Ошибка ПДП(DMA) в КШ(ВС) (%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.",
```

```
    "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 =
            extmil1553ud_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, mil1553_readDmaFixBlocks, &container);
// чтение dma блоков

            uint64_t timeEnd = getTimeStamp();
            printf("%s: processingRts [%d] - mil1553_readDma : %ld us\n",
abonent.name, index, timeEnd - timeStart);

            if(ret != GOOD_CODE) LOGGER_PRINT("%s: Ошибка mil1553_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);
```

```
                    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",
subaddress,
                                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);
```

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

    free(container.data);
}

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

    // построение массива обнаруженных девайсов
    mil1553_getArrayDevices(&devices, &devicesCount);
    for (index = 0; index < devicesCount; index++) {
        // Abonent abonent =
open_abonent_from_string(devices[index].mDev.mNameChannel);
        devices[index].mDev.mFD = mil1553_open(devices[index].mDev.mNameChannel);
        VERSION ver;
        memset(&ver, 0, sizeof(VERSION));

        if(GOOD_CODE != mil1553_getDeviceInfo(devices[index].mDev.mFD, &ver))
        {
            LOGGER_PRINT(
                        "Девайс: '%s' ОШИБКА mil1553_getDeviceInfo\n",
devices[index].mDev.mNameChannel);
        }

        unsigned int verValue = 0;
        if(GOOD_CODE != mil1553_getDriverVersion(devices[index].mDev.mFD,
&verValue))
        {
            LOGGER_PRINT(
                        "Девайс: '%s' ОШИБКА mil1553_getDriverVersion\n",
devices[index].mDev.mNameChannel);
        }

        PCI_LOCATION loc;
        memset(&loc, 0, sizeof(PCI_LOCATION));
        if(GOOD_CODE != mil1553_getDevicePciInfo(devices[index].mDev.mFD, &loc))
        {
            LOGGER_PRINT(
                        "Девайс: '%s' ОШИБКА mil1553_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);

        mil1553_close(devices[index].mDev.mFD);
        devices[index].mDev.mFD = -1;
    }
#endif
```

```
}

void readChannels(void) {
#ifndef _WIN32
    int index;
    // построение массива обнаруженных каналов
    mil1553_getArrayChannels(&channels, &channelsCount);
    for (index = 0; index < channelsCount; index++) {
        int numDev = mil1553_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: dmablk= %d; err= %d;\n", params.mNames[0],
                statTest.mCh[0].mDmaBlk, statTest.mCh[0].mErr);
    if (TRUE_VAL == configChannels.mIsReady[1])
        LOGGER_PRINT("rt1 %s: dmablk= %d; err= %d;\n", params.mNames[1],
                statTest.mCh[1].mDmaBlk, statTest.mCh[1].mErr);
    if (TRUE_VAL == configChannels.mIsReady[2])
        LOGGER_PRINT("rt2 %s: dmablk= %d; err= %d;\n", params.mNames[2],
                statTest.mCh[2].mDmaBlk, statTest.mCh[2].mErr);
    if (TRUE_VAL == configChannels.mIsReady[3])
        LOGGER_PRINT("rt3 %s: dmablk= %d; err= %d;\n", params.mNames[3],
                statTest.mCh[3].mDmaBlk, statTest.mCh[3].mErr);
    if (TRUE_VAL == configChannels.mIsReady[4])
        LOGGER_PRINT("rt4 %s: dmablk= %d; err= %d;\n", params.mNames[4],
                statTest.mCh[4].mDmaBlk, statTest.mCh[4].mErr);
    if (TRUE_VAL == configChannels.mIsReady[5])
        LOGGER_PRINT("rt5 %s: dmablk= %d; err= %d;\n", params.mNames[5],
                statTest.mCh[5].mDmaBlk, statTest.mCh[5].mErr);
    if (TRUE_VAL == configChannels.mIsReady[6])
        LOGGER_PRINT("rt6 %s: dmablk= %d; err= %d;\n", params.mNames[6],
                statTest.mCh[6].mDmaBlk, statTest.mCh[6].mErr);
    if (TRUE_VAL == configChannels.mIsReady[7])
        LOGGER_PRINT("rt7 %s: dmablk= %d; err= %d;\n", params.mNames[7],
                statTest.mCh[7].mDmaBlk, statTest.mCh[7].mErr);
    if (TRUE_VAL == configChannels.mIsReady[8])
        LOGGER_PRINT("rt8 %s: dmablk= %d; err= %d;\n", params.mNames[8],
                statTest.mCh[8].mDmaBlk, statTest.mCh[8].mErr);
}

int isValidNameChannels(void) {
    int ret = TRUE_VAL;
    int index, indexOfAllCh;
    for (index = 0; index < COUNT_CH; index++)
    {
        if (configChannels.mIsReady[index])
        {
            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",
```

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

void bcDebugProcessing(void) {
    int index = 0;  // индекс КШ
    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) &
0xFFF) * 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) & 0xFFF, (regInstrPtr.data & 0xFFF),
                    (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) & 0xFFF, (regOpDataPtr.data &
0xFFF),
                    (oper.data, 16));

        IS_EXT_WRAPPER(abonent, mil1553_close);
    }
}
```

```
#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(ВС) Чтение прерываний", &intBuffer, intBuffer.blocks);

    Abonent abonent = open_abonent_from_string(params.mNames[0]);
    do {
        count = IS_EXT_WRAPPER(abonent,
mil1553ud_getCountInterruptBlocksReadyRead);
        intBuffer.count_blocks = MAX_INTERRUPT_BLOCKS;
        unsigned int result = IS_EXT_WRAPPER(abonent,
mil1553ud_readInterruptBlocks, &intBuffer);
        if (result != GOOD) {
            LOGGER_PRINT("%s\n", "Ошибка функции mil1553ud_readInterruptBlocks");
        }
        countReadedBlocks += intBuffer.count_blocks;
    } while (count > 0);
    IS_EXT_WRAPPER(abonent, mil1553_close);

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

#ifndef _WIN32
    countDevices = mil1553_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
```

```
    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();  // ининициализация 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;
}
```

ПРИЛОЖЕНИЕ Б (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 mil1553dev-0-ch-1     //КШ
rt1 mil1553dev-0-ch-0    //ОУ1
rt2 mil1553dev-0-ch-2    //ОУ2
rt3 mil1553dev-0-ch-3    //ОУ3
rt4 mil1553dev-usb-0-ch-0 //ОУ4
rt5 192.168.10.10:19191:0 //ОУ5
rt6 192.168.10.10:19191:1 //ОУ6
rt7 NULL    //ОУ7
rt8 NULL    //ОУ8

PARAMS: duration_h= 8, duration_m= 5, line= 0, bc= /dev/mil1553dev-0-ch-1,
 rt1= /dev/mil1553dev-0-ch-0,
 rt2= /dev/mil1553dev-0-ch-2,
 rt3= /dev/mil1553dev-0-ch-3,
 rt4= /dev/mil1553dev-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!
```

```
parsing './mem/consol_inst_ram.mem':
 - End parsing!
parsing './mem/consol_op_ram.mem':
 - End parsing!
./mem/consol_inst_ram.mem was writed.
./mem/consol_op_ram.mem was writed.
./mem/consol_dat_ram.mem was writed.
Wait for test has finished, please...
--- --- ---
Test has finished
bc /dev/mil1553dev-0-ch-1: dmablk= 142316289; err= 0;
rt1 /dev/mil1553dev-0-ch-0: dmablk= 29961324; err= 0;
rt2 /dev/mil1553dev-0-ch-2: dmablk= 29961324; err= 0;
rt3 /dev/mil1553dev-0-ch-3: dmablk= 29961324; err= 0;
rt4 /dev/mil1553dev-usb-0-ch-0: dmablk= 29961324; err= 0;
rt5 /dev/192.168.10.10:19191:0: dmablk= 29961324; err= 0;
rt6 /dev/192.168.10.10:19191:1: dmablk= 29961324; err= 0;
rt7 NULL: dmablk= 0; err= 0;
rt8 NULL: dmablk= 0; err= 0;
--- --- ---
Time spent on the test = 29100 seconds
```

Рисунок Б1 - Пример визуализации тестовой программы

СПИСОК СОКРАЩЕНИЙ

ПО – программное обеспечение;

МКИО – интерфейс по ГОСТ Р 52070-2003;

КШ – контроллер шины;

ОУ – оконечное устройство;

МШ – монитор шины;

МША – монитор шины адресный;

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

Лист регистрации

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