Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент Брянская Екатерина Вадимовна

фамилия, имя, отчество

Группа ИУ7-62Б

Тип практики производственная

Название предприятия АО "Московский научно-исследовательский институт 'АГАТ' "

Студент _____ Брянская Е.В.
подпись, дата *фамилия, и.о.*

Руководитель практики _____ Толпинская Н.Б.
подпись, дата *фамилия, и.о.*

Руководитель предприятия _____ _____
подпись, дата *фамилия, и.о.*

Оценка _____

2021 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

З А Д А Н И Е
по производственной практике

Студент группы ___ИУ7-62Б_____

_____Брянская Екатерина Вадимовна_____
(Фамилия, имя, отчество)

Тема практики

Разработка и отладка библиотеки алгоритмов взаимодействия персонального компьютера (ПК) и контрольной измерительной аппаратурой радиолокационной головкой самонаведения (КИА РГС).

Исходные данные и решаемые задачи

На основании руководства по программированию и протокола КМБО (канала межблочного обмена) разработать библиотеку алгоритмов, которая должна обеспечить информационное взаимодействие персонального компьютера и контрольной измерительной аппаратурой РГС по платам:

____- PCI-КМБО,

____- USB-КМБО,

____- Ethernet-КМБО.

Оформление отчёта:

Отчёт на _20-30_ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

_____

_____

_____

Дата выдачи задания « ___ » _____ 20__ г.

Руководитель практики от предприятия _____ _____
(Подпись, дата) (И.О.Фамилия)

Руководитель практики от МГТУ им. Н.Э. Баумана_____ __Толпинская Н.Б._
(Подпись, дата) (И.О.Фамилия)

Студент _____ __Брянская Е.В._____
(Подпись, дата) (И.О.Фамилия)

Отзыв о прохождении производственной практики

Студент группы ИУ7-62 Екатерина проходила производственную практику в НИО-5 лаборатории №502 по теме “Разработка и отладка библиотеки алгоритмов взаимодействия персонального компьютера и контрольной измерительной аппаратурой РГС”. За время прохождения производственной практики студент группы ИУ7-62 Екатерина разработала библиотеку алгоритмов взаимодействия ПК и КИА РГС в соответствии с руководством программирования и протоколом КМБО для плат PCI-КМБО, USB-КМБО и Ethernet-КМБО. Библиотека была реализована в среде программирования Microsoft Visual Studio C# 2015.

Библиотека алгоритмов взаимодействия ПК и КИА РГС будет использована в программном обеспечении проверки изделия РГС.

Практика выполнена в полном объеме и соответствует программе практики, а представленный отчет выполнен на высоком уровне. По итогам практики студент группы ИУ7-62 Екатерина заслуживает оценки «отлично».

Начальник НИО5
АО «МНИИ «Агат»_____ Вексин С.И.
(Подпись, дата, печать)

Содержание

Введение

Головка самонаведения (ГСН) — автоматическое устройство, которое устанавливается на управляемое средство поражения (ракету, бомбу, торпеду и др.) для обеспечения прямого попадания в объект атаки или сближение на расстояние, меньшее радиуса поражения боевой части средства поражения (СП), то есть для обеспечения высокой точности наведения на цель. ГСН является элементом системы самонаведения. [1]

СП, оборудованное ГСН, может «видеть» «подсвеченную» носителем или ей самой, излучающую или контрастную цель и самостоятельно наводиться на неё, в отличие от ракет, наводимых командным способом.

Виды ГСН:

- **РГС (РГС, РЛГСН) — радиолокационная ГСН**;
- ТГС (ИКГСН) — тепловая, инфракрасная ГСН;
- ТВГСН — телевизионная ГС;
- ТПВГС - тепловизионная ГС;
- ЛГСН - Лазерная ГС.

Канал межблочного обмена (КМБО) может быть использован для обмена данными между управляющей ЭВМ и внешними устройствами (в частности, НЧ и СВЧ аппаратурой) в режиме реального времени. Техническим результатом является повышение точности синхронизации обмена данными. [2]

Цель работы - разработать и отладить библиотеку алгоритмов взаимодействия персонального компьютера (ПК) и контрольной измерительной аппаратурой радиолокационной головкой самонаведения (КИА РГС).

Выделены следующие **задачи**:

1) ознакомиться с алгоритмами взаимодействия ПК и КИА РГС;
2) изучить протокол канала межблочного обмена (КМБО);
3) проработать особенности работы с PCI, USB, Ethernet;
4) разработать соответствующую библиотеку алгоритмов и отладить её.

Основная часть

1 Общие положения

1.1 Канал межблочного обмена (КМБО)

1.1.1 Основные характеристики канала

Скорость передачи данных - 1000 Кбит/сек.

Расстояние передачи – до 10 м.

1.1.2 Состав канала

В состав канала обмена входят:

- ведущая станция – ПК с контроллером;
- оконечные устройства – модули блоков аппаратуры.

При необходимости к каналу может подключаться вспомогательное (сервисное) оборудование на правах оконечных устройств. Общее количество устройств в составе канала:

- контроллер – 1;
- оконечные устройства – до 32.

Ведущей станцией, управляющей обменом, является **контроллер**. **Оконечные устройства** осуществляют прием информации от контроллера или передачу информации контроллеру под управлением контроллера.

Для организации линии связи между контроллером и оконечными устройствами используются две двухпроводные магистральные линии:

- линия передачи тактового сигнала (SYN);
- линия передачи данных (DATA).

Взаимный обмен информацией между контроллером и оконечными устройствами осуществляется **кадрами информации**.

1.1.3 Форматы кадров информационного обмена

Форматы кадров разделяется на два вида:

- формат кадра передачи информации от контроллера (кадр W);
- формат кадра приема информации от оконечного устройства (кадр R).

Формат кадров W и R одинаковый и имеет вид:

ЗАГ	ИС	ИС	...	ИС	БЧ

где:

ЗАГ – заголовок кадра (16 бит);

ИС – информационное слово (16 бит). Количество информационных слов в кадре – от одного до 64;

БЧ – бит четности (1 бит).

При формировании кадров W на линии данных контроллер является активным на протяжении всего кадра. По окончании кадра контроллер переводится в состояние паузы.

При формировании кадров R на линии данных контроллер является активным на протяжении заголовка кадра. По окончании заголовка контроллер переводится в состояние паузы. На протяжении части кадра, соответствующей ИС и БЧ, на линии данных активным является оконечное устройство.

По линии тактового сигнала контроллер является активным как во время кадра (W или R), так и в паузе между кадрами. Оконечные устройства по линии тактового сигнала работают только на прием.

Формат заголовка кадра имеет вид:

АДР	W/R	ЧС	БЗЧ

где:

АДР – адрес оконечного устройства (8 бит);

W/R – признак Передача - Чтение (1 бит);

ЧС – число информационных слов в кадре (6 бит);

БЧЗ – бит четности заголовка (1 бит).

Адрес оконечного устройства – положительное двоичное число от 1 до 255. Передача адреса начинается со старшего разряда и кончается младшим разрядом.

Признак W/R – 1 бит, имеет значение 0 – передача данных от контроллера оконечному устройству (кадр W), 1 – прием данных от оконечного устройства (кадр R).

Число информационных слов в кадре – положительное двоичное число от 1 до 64. Передача ЧС начинается со старшего разряда и кончается младшим разрядом. Код 000000 соответствует числу слов 64.

Бит четности заголовка дополняет заголовок до нечетности. [3]

Числовая информация передается дополнительным кодом.

1.1.4 Требования к контроллеру

Контроллер, как правило, является устройством, управляемым программно процессором ПК. Может находиться в двух состояниях:

- состояние паузы;
- состояние работы.

В **состоянии паузы** контроллер находится в перерыве между кадрами передачи или приема информации.

В **состоянии работы** контроллер осуществляет прием или передачу заданного числа слов.

При передаче данных (кадр W) контроллер передает заголовок, заданное число слов и бит четности БЧ. Бит четности вычисляется контроллером аппаратно. После окончания передачи кадра контроллер переходит в состояние паузы.

При приеме данных (кадр R) контроллер передает заголовок, после чего принимает заданное число слов и бит четности БЧ. Проверка соответствия бита четности проводится контроллером аппаратно. После окончания приема кадра контроллер переходит в состояние паузы.

При передаче кадра W, а также при передаче заголовка кадра R контроллер одновременно производит чтение с линии данных, которые он сам передал, и аппаратно

производит сравнение переданных и принятых данных. Результат (1 бит, 0 - при совпадении, 1 - при несовпадении) должен быть доступен для чтения процессору по окончании кадра.

1.1.5 Требования к оконечному устройству

Оконечное устройство, как правило, является аппаратным устройством и функционирует в соответствии с заложенным в него алгоритмом. При этом оконечное устройство может находиться в 4-х состояниях:

- исходное состояние;
- состояние приема заголовка кадра;
- состояние ожидания паузы;
- состояние работы.

После приема заголовка кадра (16 бит) устройство сравнивает адрес АДР в заголовке с собственным адресом.

При совпадении адреса и правильном бите четности заголовка БЧЗ оконечное устройство переходит в состояние работы. При этом в зависимости от бита W/R осуществляется прием данных с линии или выдача данных на линию. Устройство принимает или передает число слов, на которое оно настроено аппаратно. По окончании приема или передачи указанного числа слов и бита четности БЧ устройство переходит в состояние ожидания паузы.

Характер исполнения принятой устройством информации при несовпадении числа слов в кадре и аппаратно заложенного в устройство числа слов, а также при неправильном бите четности кадра БЧ настоящим протоколом не определяется и устанавливается в каждом случае отдельно.

При несовпадении адреса или неправильном бите четности заголовка БЧЗ оконечное устройство переходит в состояние ожидания паузы.

1.2 USB-КМБО

1.2.1 Общее

Контроллер реализован в виде внешнего USB модуля с размерами 30мм × 85мм × 145мм и предназначен для сопряжения персонального компьютера (ПК) через порт USB

с каналом межблочного обмена (КМБО) и каналами разовых (битовых) команд (РК) и используется в составе автоматизированной контрольно-испытательной аппаратуры (АКИА).

В отличие от контроллера PCI-КМБО контроллер USB-КМБО имеет ряд дополнительных функций, позволяющих производить проверку работы аппаратуры при предельных отклонениях временных параметров сигналов КМБО.

Контроллер включает в себя ПЛИС (программируемая логическая интегральная схема), реализующую протокол обмена КМБО, приемо-передатчики сигналов КМБО и интерфейсное устройство шины USB (микросхема FT2232H). [4]

FTDI – драйвер и библиотека функций микросхемы FT2232H, используемой в контроллере USB-КМБО. [5]

1.2.2 Состав изделия

В состав изделия входят следующие функциональные узлы.

1. **ППЗУ** - перепрограммируемое постоянное запоминающее устройство (EEPROM), предназначено для хранения данных конфигурации изделия.
2. **Интерфейс ППЗУ** обеспечивает чтение ППЗУ и его программирование по шине USB.
3. **Интерфейс USB** обеспечивает протокол обмена изделия с шиной USB.
4. **Кварцевый генератор с умножителем** обеспечивает формирование тактовых импульсов с частотами 6, 12 и 48 МГц.
5. **Приемный буфер RX** предназначен для приема информации, поступающей с шины USB, с последующей передачей ее в узлы микросхемы EPM9320LC84.
6. **Передающий буфер TX** предназначен для передачи информации в шину USB, поступающей от узлов микросхемы EPM9320LC84.
7. **Регистр выходных РК** обеспечивает прием и хранение разовых команд РК, поступающих с шины USB в приемный буфер RX.
8. **Регистр командного слова** обеспечивает прием и дешифрацию командного слова, поступающего с шины USB в приемный буфер RX.
9. **Интерфейс КМБО** обеспечивает формирование W-кадров и заголовков R-кадров информации КМБО, значения которых считываются из приемного буфера RX, а также прием данных R-кадров и запись их в передающий буфер TX. Устройство

осуществляет также формирование бита четности в кадрах W, контроль бита четности в кадрах W и R (ВН) и т.д.

10. **Регистр входных РК** обеспечивает передачу входных разовых команд РК в передающий буфер ТХ.

11. **Твердотельные реле** предназначены для выдачи выходных РК.

12. **Приемопередатчики RS-485** обеспечивают формирование уровней сигналов КМБО.

1.3 PCI-КМБО

1.3.1 Основные части

Изделие состоит из следующих функциональных устройств.

1. **Интерфейс PCI** обеспечивает связь изделия с шиной PCI ПЭВМ. Обмен информацией между изделием и ПЭВМ осуществляется через четыре порта ввода/вывода и один канал прерывания.

2. **Конфигурационная память** выполняет координационные функции по распределению ресурсов между устройствами при помощи специальной конфигурационной программы операционной системы ПЭВМ. После загрузки операционной системы в соответствующих регистрах конфигурационной памяти можно прочитать выделенные изделию базовый адрес ввода/вывода и номер канала прерывания.

3. **Буферное запоминающее устройство (БЗУ)** обеспечивает хранение передаваемых и принимаемых данных КМБО. БЗУ двухпортовое, через первый порт происходит обмен информацией между БЗУ и ПЭВМ, через второй порт – между БЗУ и КМБО. Обмен между БЗУ и КМБО происходит во время передачи или приема кадров информации КМБО, а между БЗУ и ПЭВМ - в паузах между кадрами.

4. **Счетчик адреса** обеспечивает последовательные запись или чтение информации в/из БЗУ.

5. **Регистры**, предназначенные для управления изделием со стороны ПЭВМ.

6. **Устройство синхронизации** обеспечивает синхронную работу составных узлов изделия и формирование структуры кадра.

7. **Устройство формирования сигналов КМБО** обеспечивает формирование W-кадров и заголовков R-кадров информации КМБО, значения которых хранятся в БЗУ, а также прием данных R-кадров и запись их в БЗУ. Устройство осуществляет

также формирование бита четности в кадрах W, контроль бита четности в кадрах W и R (ВН) и контроль встречной работы (ВР).

8. **Формирователь прерываний** обеспечивает формирование сигнала прерываний от таймера и по окончании передачи/приема кадров КМБО.

9. **Таймер** обеспечивает формирование импульсов с программируемым периодом следования.

10. **Счетчик времени**

1.3.2 Режимы изделия

Контроллер может находится в следующих режимах.

1. **Режим ВЫКЛ**
   В этом режиме через конфигурационные регистры производится проверка наличия подключенного изделия, чтение базового адреса и номера прерывания, назначенные операционной системой ПЭВМ (персональной электронной вычислительной машиной), и перевод изделия в режим ПОКОЙ.

2. **Режим ПОКОЙ**
   Отличается от режима ВЫКЛ тем, что появляется возможность записи/чтения рабочих регистров изделия для установки режимов его работы.

3. **Режим КОНТРОЛЛЕР**
   Включается программно. В этом режиме изделие функционирует в режиме ведущей станции (контроллера) и обеспечивает следующие функции:

   - обмен информацией по линии связи, скорость задается программно;
   - хранение передаваемых и принимаемых данных в БЗУ (буферное запоминающее устройство) емкостью в 64 шестнадцатиразрядных слов;
   - формирование кадров передачи данных (кадры W) и приема данных (кадры R) длиной от 1 до 64 информационных слов;
   - аппаратное формирование бита четности в кадрах W и аппаратный контроль бита четности в кадрах W и R;
   - аппаратное побитное сравнение передаваемых и принимаемых данных;
   - формирование сигнала прерывания в конце каждого кадра.

1.4 Ethernet-КМБО

1.4.1 Общее

МС – устройство, сопрягающее персональный компьютер ПК, активный канал и канал радиокоррекции (РК).

Применяется в составе аппаратуры контроля головки. Назначение модуля – преобразование интерфейсов КМБО и МКИО, контроль источников напряжения, выдача битовых команд. МС соединятся с ПК по Ethernet интерфейсу, с активным каналом по МКИО, с каналом РК по каналу межблочного обмена (КМБО). [6]

1.5 Вывод

Необходимо учесть все особенности работы с платами PCI-КМБО, USB-КМБО, Ethernet-КМБО для того, чтобы разработать необходимую библиотеку для информационного взаимодействия ПК и измерительной аппаратуры.

2 Конструкторский раздел

В этом разделе будут подробно описаны основные функции библиотеки, приведены схемы их работы.

2.1 Общий алгоритм

Общий алгоритм выглядит следующим образом (рисунок 2.1):

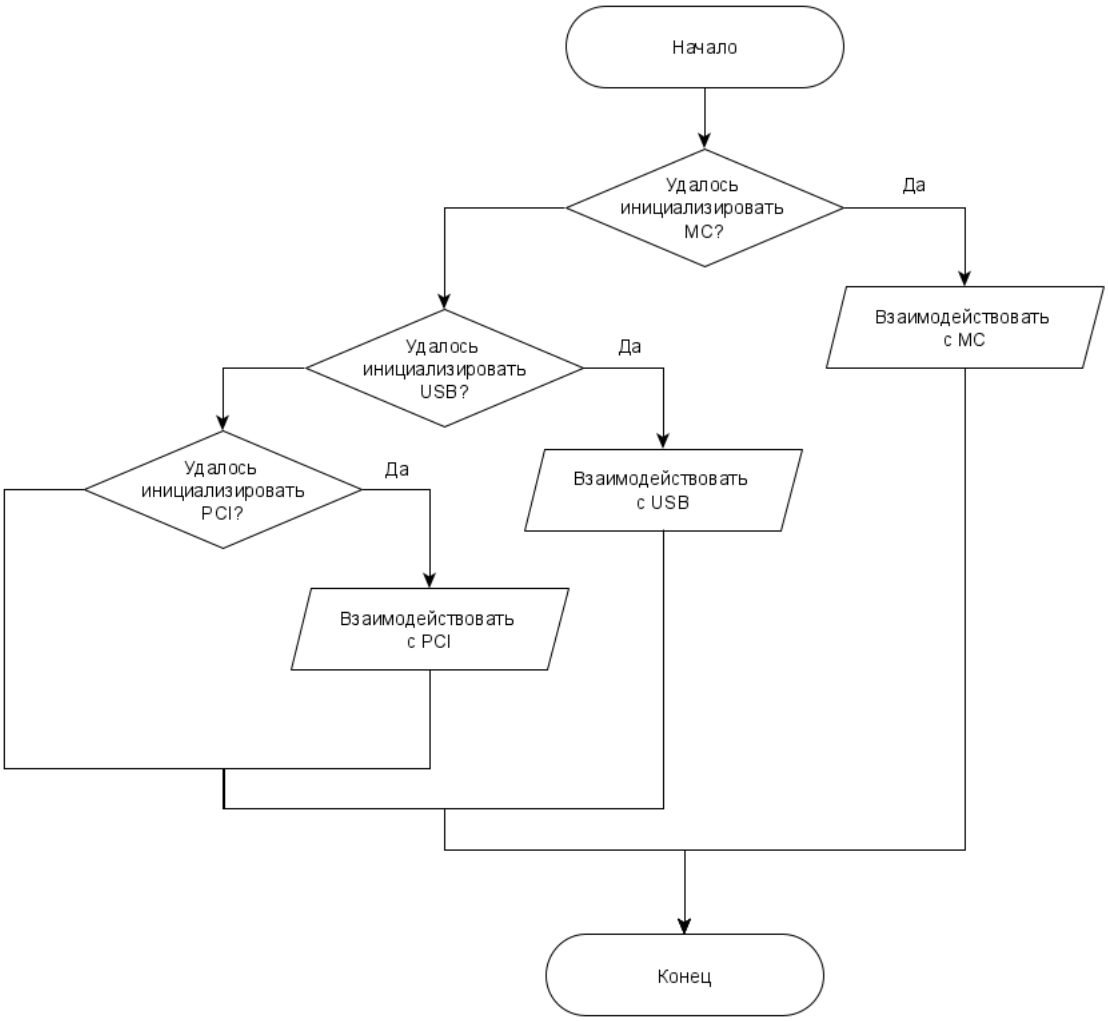

Рисунок 2.1 – Общий алгоритм

2.2 USB

2.2.1 Функция инициализации (USB_Open)

Схема изображена на рисунке 2.2.

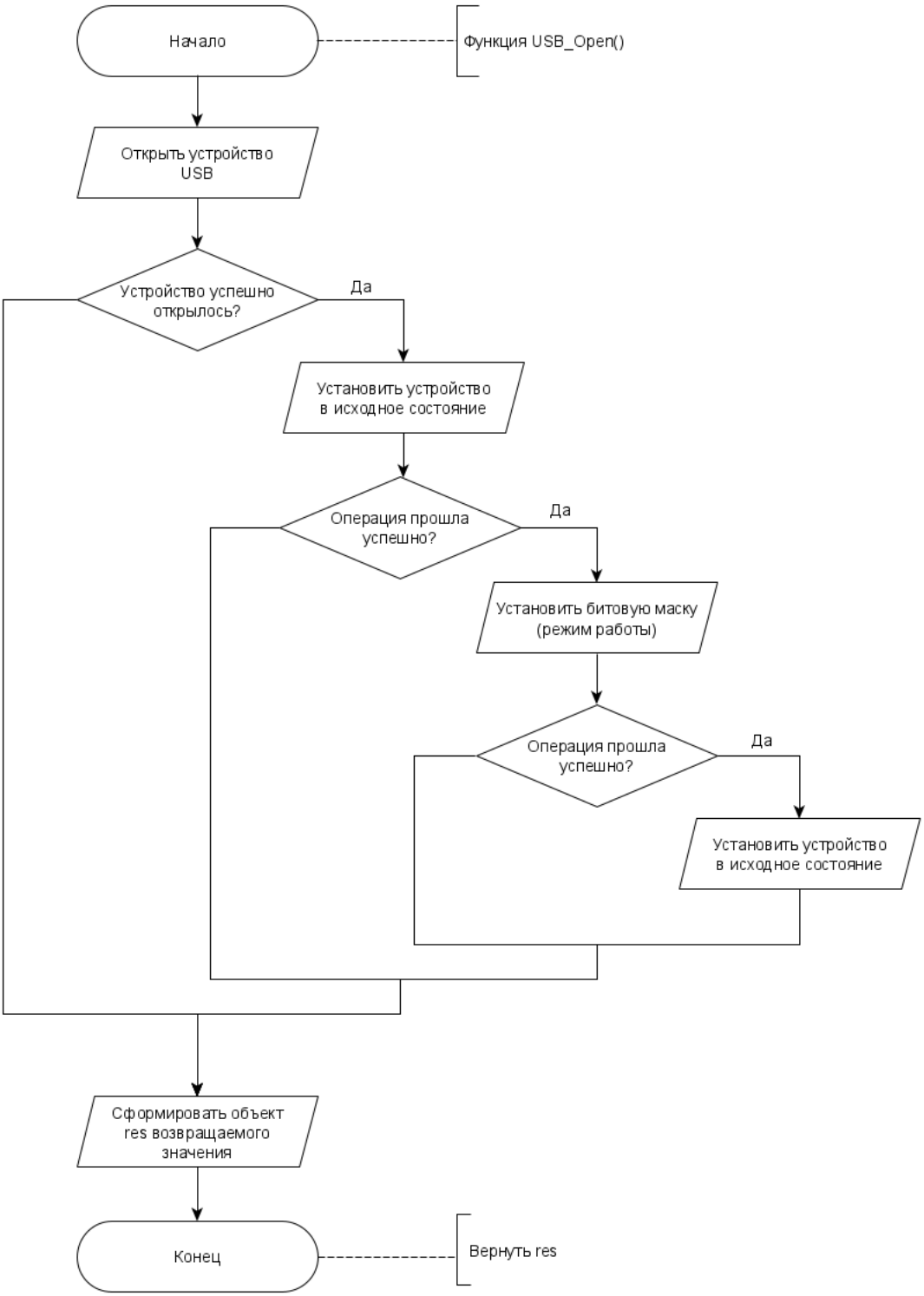

Рисунок 2.2 – USB_Open()

2.2.2 Прекращение работы с USB (USB_Close)

Схема изображена на рисунке 2.3.

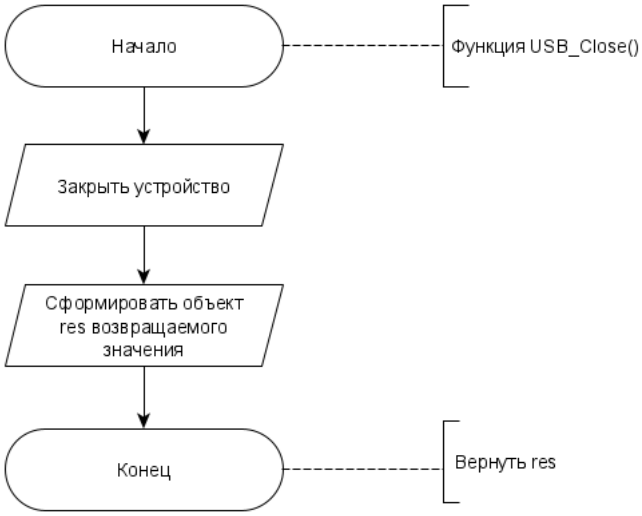

Рисунок 2.3 – USB_Close()

2.2.3 Функция записи данных в устройство USB-КМБО (USB_Write)

Схема изображена на рисунке 2.4.

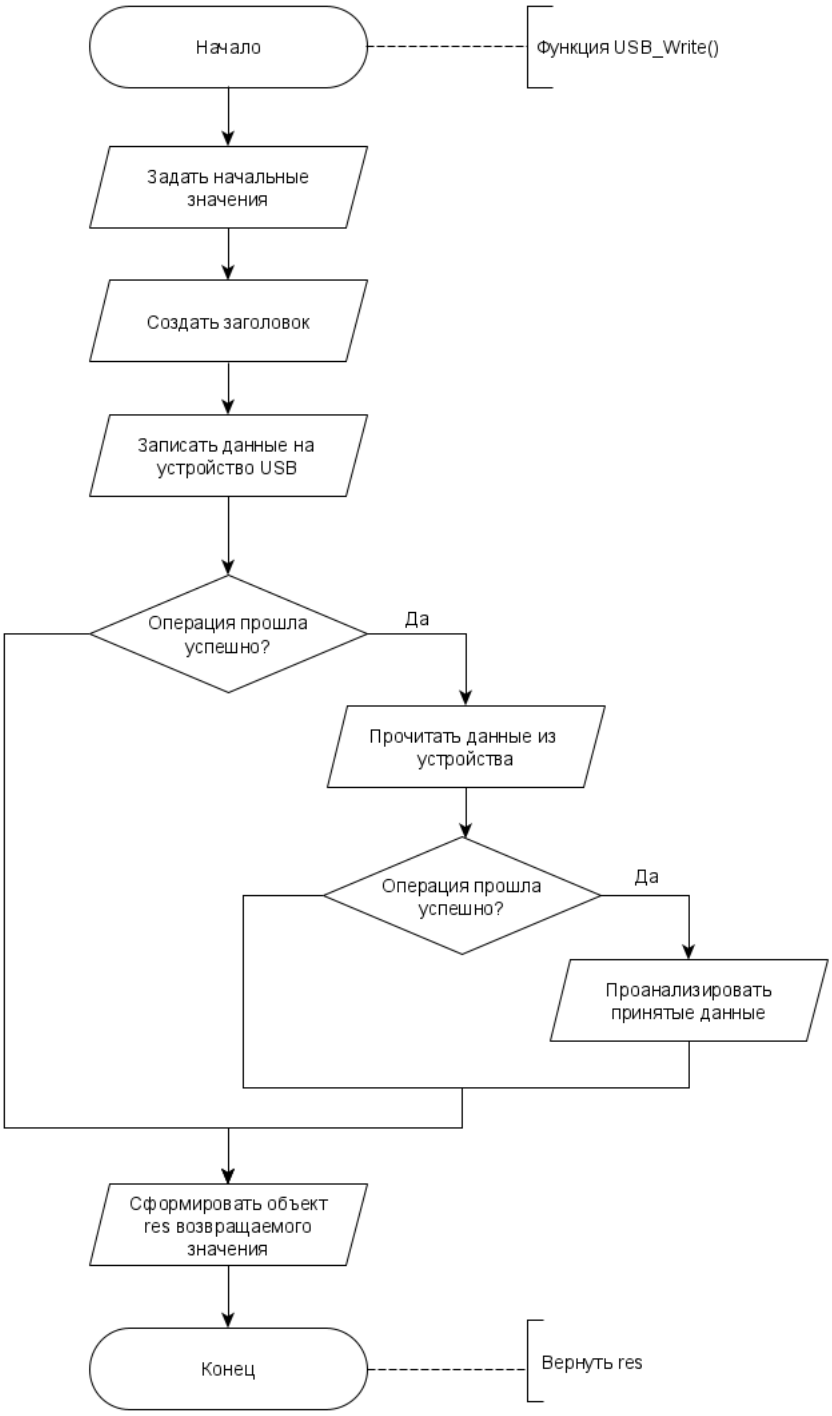

Рисунок 2.4 – USB_Write()

2.2.4 Функция чтения данных из устройства USB-КМБО (USB_Read)

Схема изображена на рисунке 2.5.

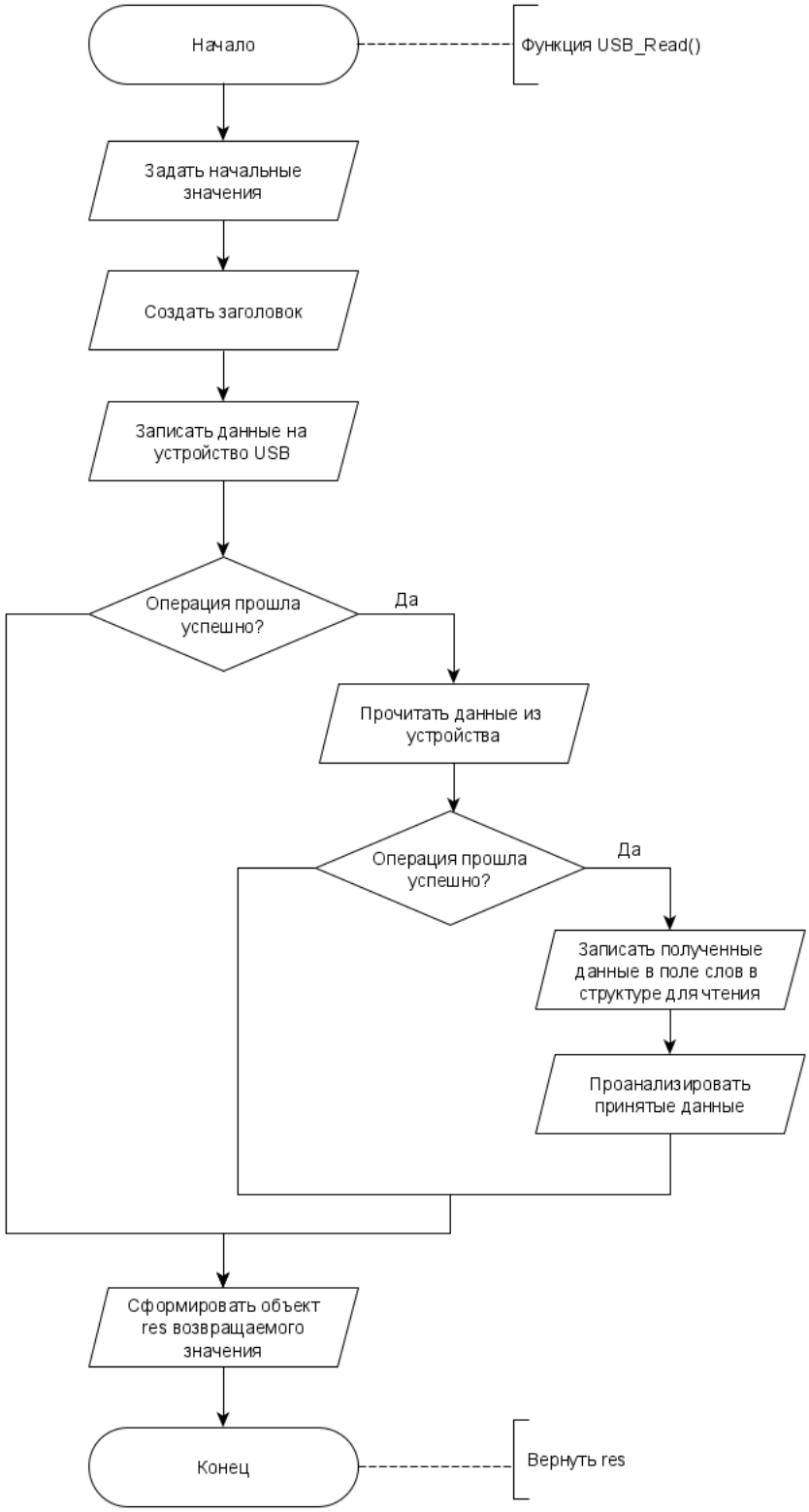

Рисунок 2.5 – USB_Read()

2.3 PCI

2.3.1 Функция инициализации (PCI_Open)

Схема изображена на рисунке 2.6.

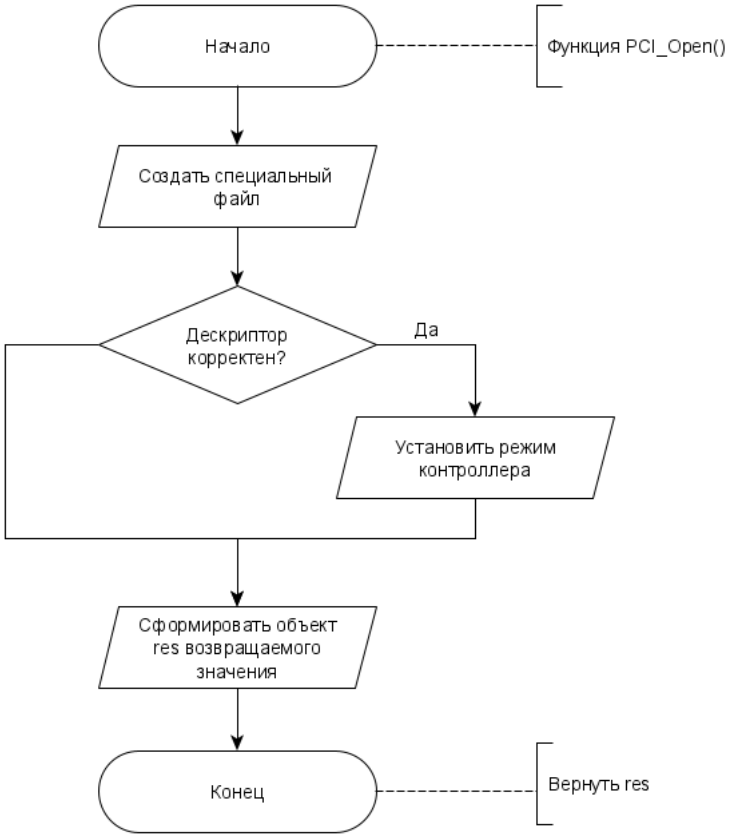

Рисунок 2.6 – PCI_Open()

2.3.2 Прекращение работы с PCI (PCI_Close)

Схема изображена на рисунке 2.7.

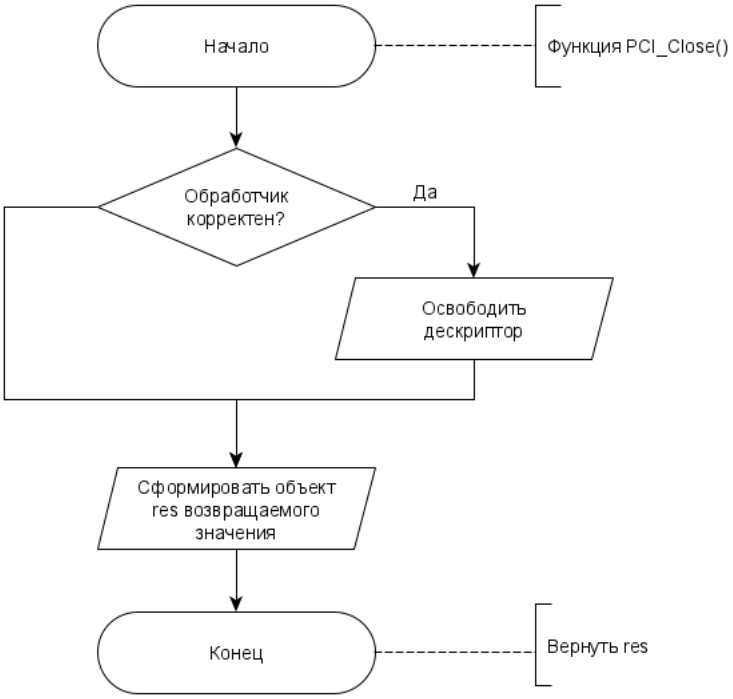

Рисунок 2.7 – PCI_Close()

2.3.3 Функция записи данных в устройство PCI-КМБО (PCI_Write)

Схема изображена на рисунке 2.8.

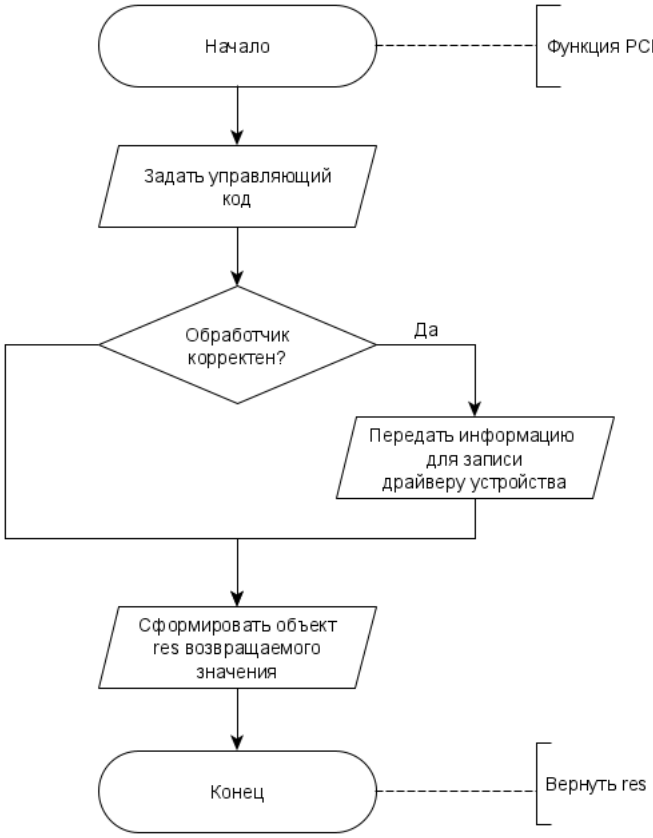

Рисунок 2.8 – PCI_Write()

2.3.4 Функция чтения данных из устройства PCI-КМБО (PCI_Read)

Схема изображена на рисунке 2.9.

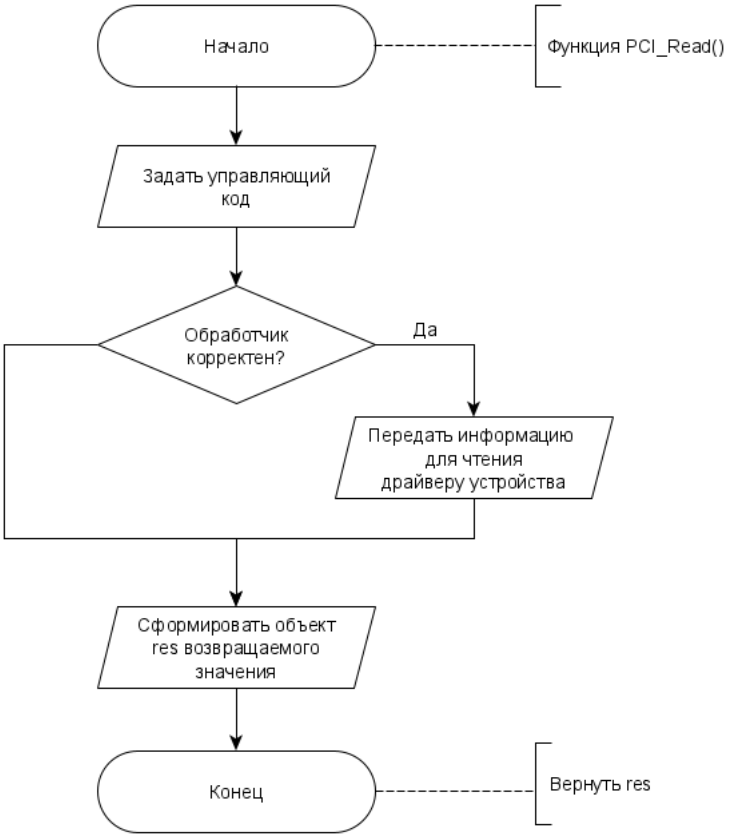

Рисунок 2.9 – PCI_Read()

2.4 Ethernet-КМБО

2.4.1 Функция инициализации (MC_Open)

Схема изображена на рисунке 2.10.

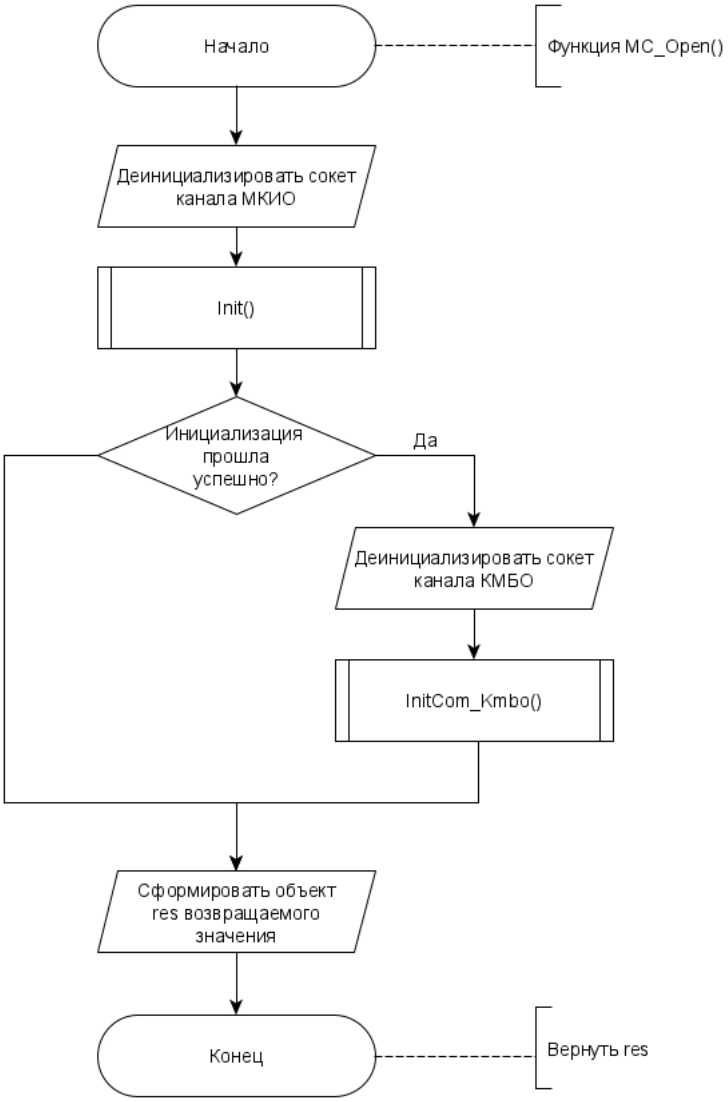

Рисунок 2.10 – MC_Open()

2.4.2 Прекращение работы с МС (MC_Close)

Схема изображена на рисунке 2.11.

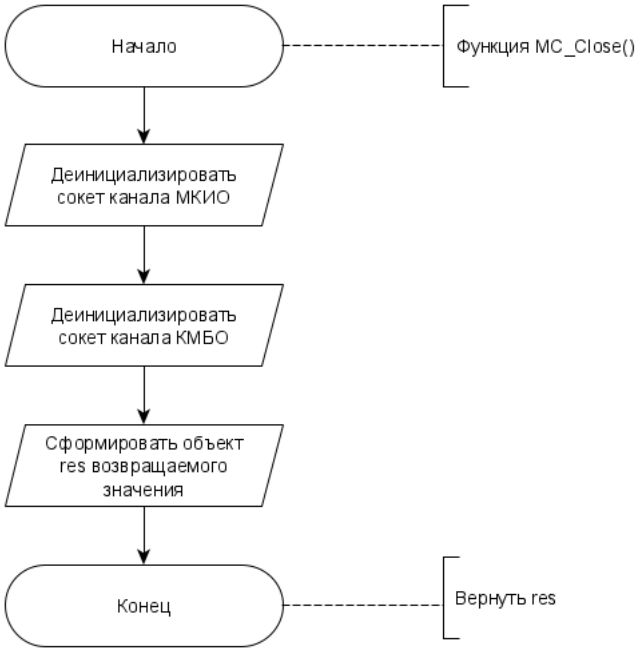

Рисунок 2.11 – MC_Close()

2.4.3 Функция записи данных в устройство МС-КМБО (MC_Write)

Схема изображена на рисунке 2.12.

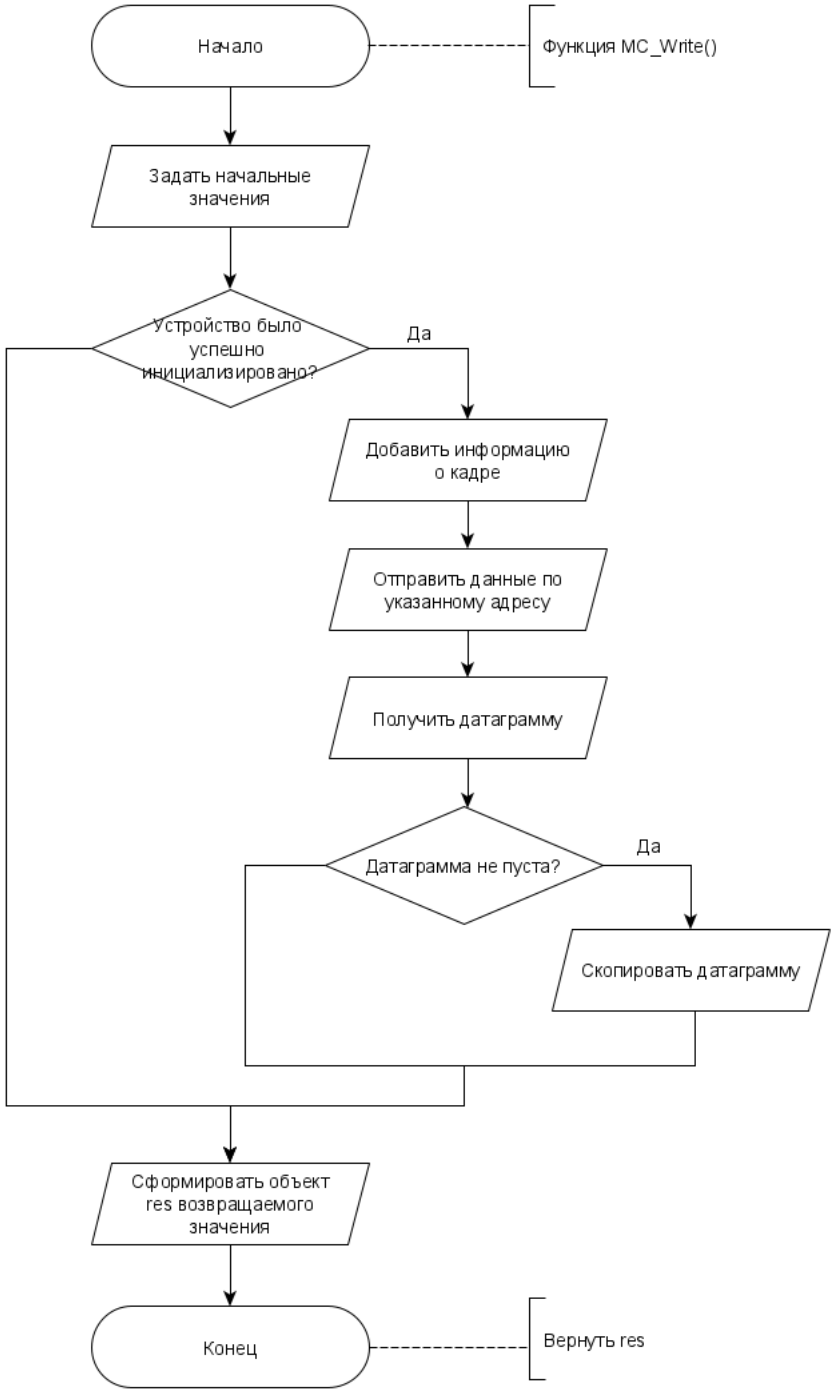

Рисунок 2.12 – MC_Write()

2.4.4 Функция чтения данных из устройства МС-КМБО (MC_Read)

Схема изображена на рисунке 2.13.

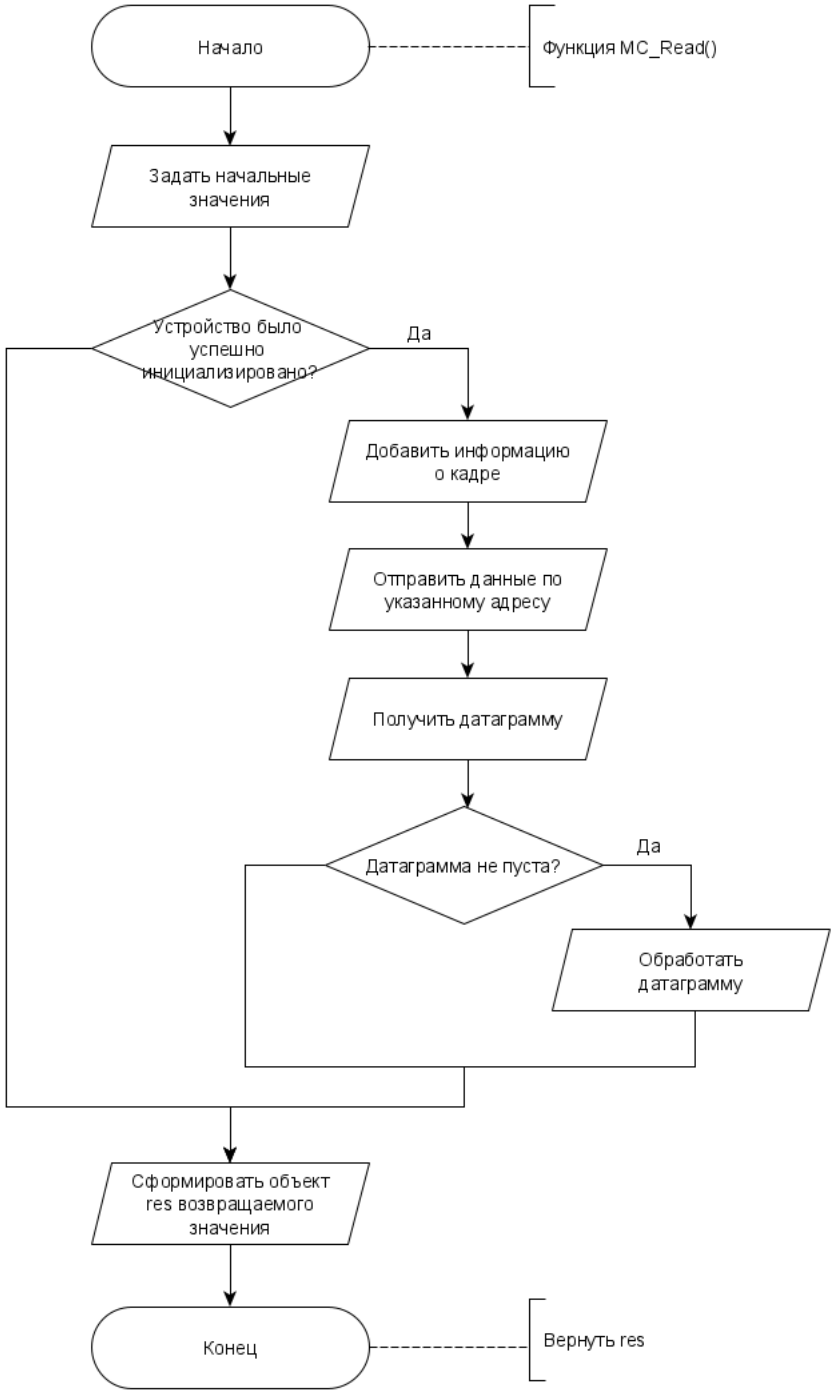

Рисунок 2.13 – MC_Read()

2.4.5 Функция инициализации сокета МКИО (Init)

Схема изображена на рисунке 2.14.

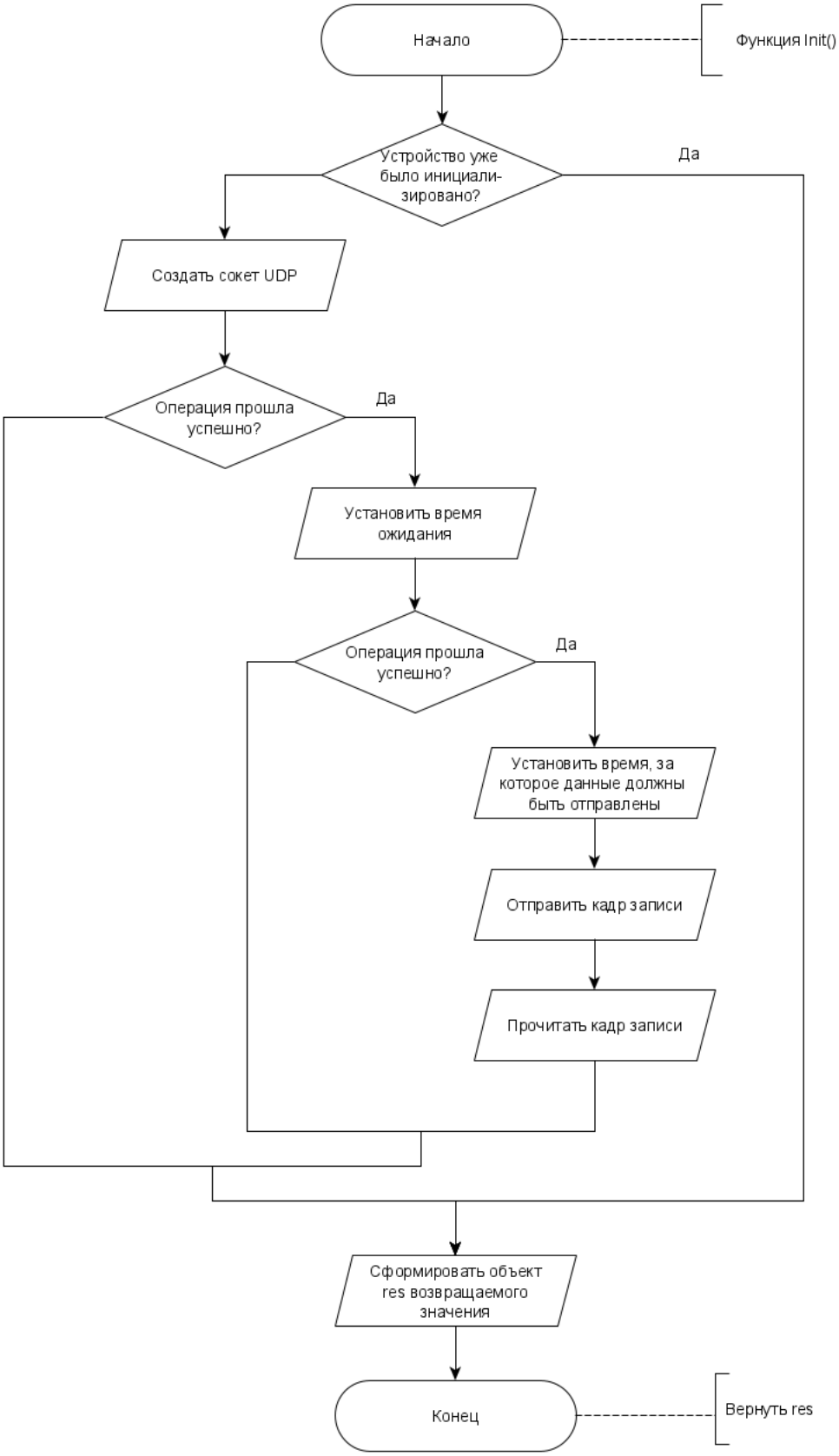

Рисунок 2.14 – Init()

2.4.6 Функция инициализации сокета КМБО (InitCom_Kmbo)

Схема изображена на рисунке 2.15.

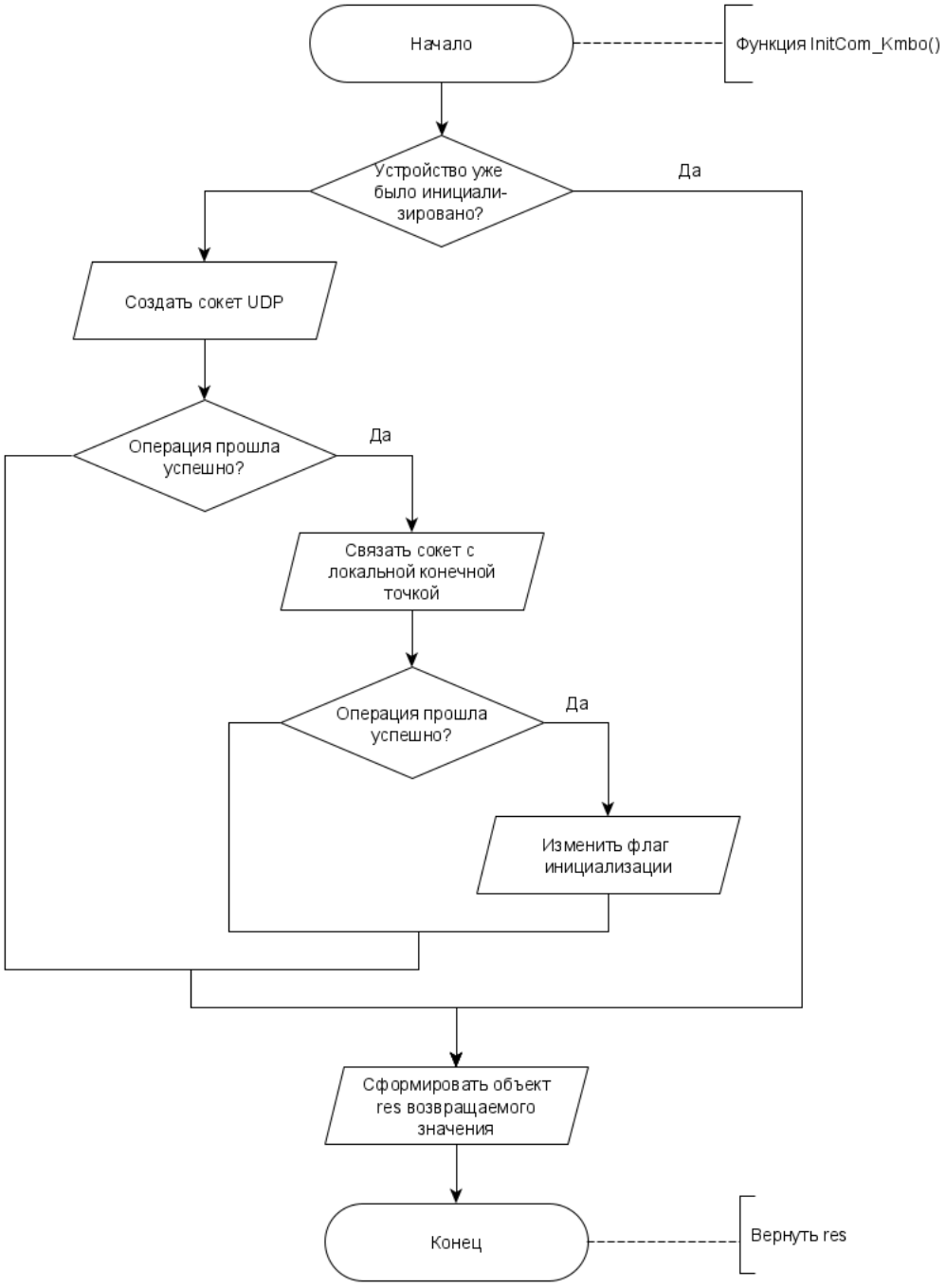

Рисунок 2.15 – InitCom_Kmbo()

2.5 Вывод

В разделе приведены схемы основных функций, входящих в состав библиотеки.

3 Технологический раздел

3.1 Язык программирования

При разработке программного продукта был использован язык программирования C#. [7] В качестве среды разработки была использована Visual Studio 2015. [8] Данный выбор обусловлен прежде всего тем, что именно эти инструменты привлекаются для разработки ПО на предприятии.

Такой язык программирования был выбран в качестве основного на предприятии по нескольким причинам:

1) большое количество готовых библиотек и шаблонов;

2) исчерпывающая документация;

3) низкий порог вхождения;

4) поддержка ООП.

3.2 Используемые классы

На рисунке 3.1 представлена UML-диаграмма классов.

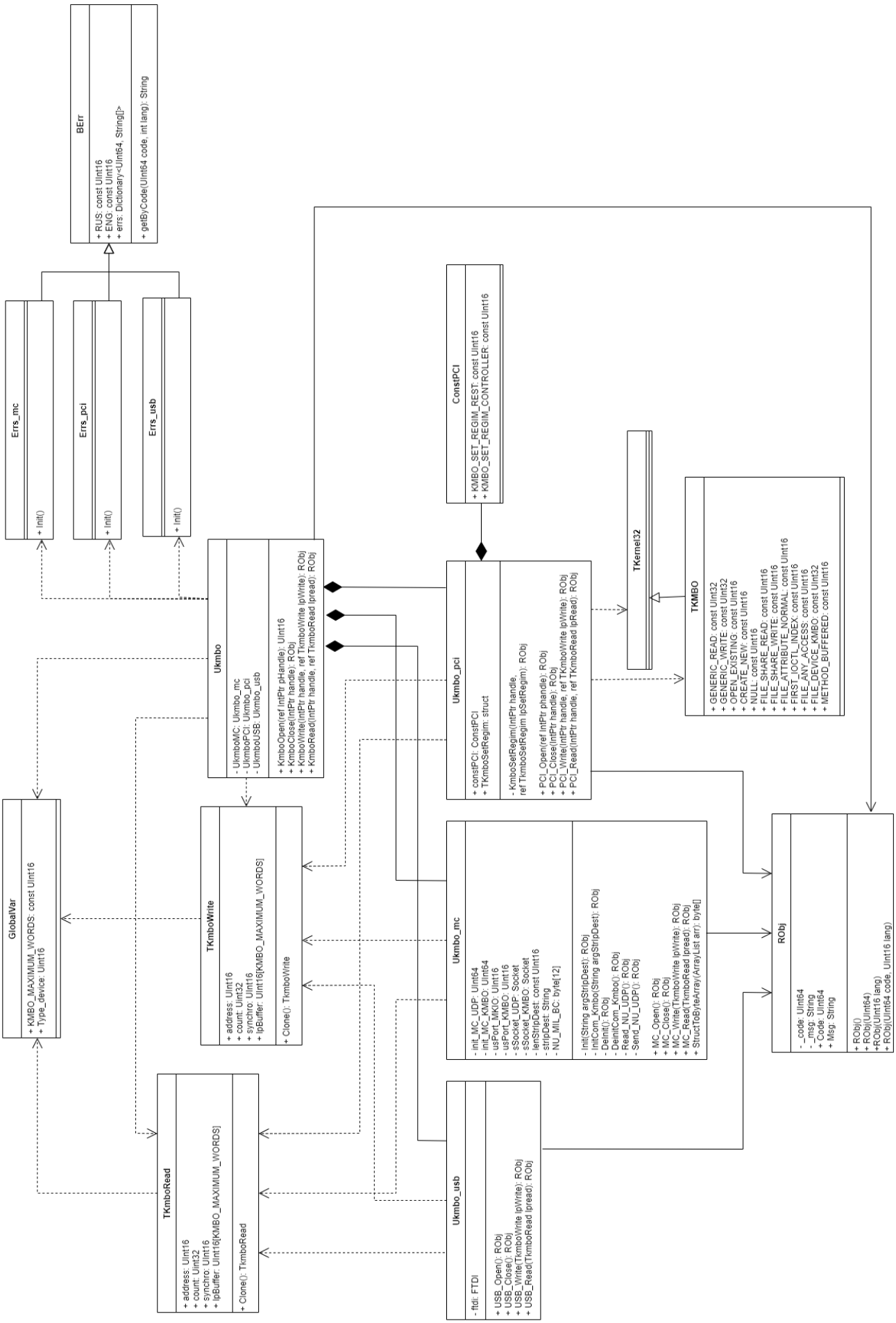

Рисунок 3.1 – Используемые классы

3.3 Используемые классы

1. **Class GlobalVar**

   Класс глобальных переменных.

2. **Class RObj**

   Класс возвращаемого значения, хранит код ошибки и соответствующую информацию, заполнение/изменение этого поля происходит одновременно с заданием/изменением поля кода ошибки.

3. **Class BErr**

   Базовый класс ошибки. Содержит метод getByCode(), который по коду ошибки находит соответствующую информацию о ней, сначала поиск осуществляется по заранее определённому словарю ошибок, в случае неудачи происходит обращение к WinAPI.

4. **Class Errs_mc**

   Класс ошибок, определённых для МС.

5. **Class Errs_usb**

   Класс ошибок, определённых для USB.

6. **Class Errs_pci**

   Класс ошибок, определённых для PCI.

7. **Class TKMBO**

   Содержит необходимые для работы константы.

8. **Class TKmboRead**

   Класс данных для чтения.
   Содержит поля адреса, количества записываемых слов, флаг, который определяет будет ли выводиться сигнал на осциллограф или нет и массив записанных слов.

9. **Class TKmboWrite**

   Класс данных для записи.
   Содержит поля адреса, количества записываемых слов, флаг, который определяет будет ли выводиться сигнал на осциллограф или нет и массив записанных слов.

10. **Class Ukmbo**

    Содержит основные методы взаимодействия с КМБО, такие как Open, Close, Read, Write.

11. **Class Ukmbo_mc**

    Содержит методы для взаимодействия с MC.

12. **Class Ukmbo_usb**

    Содержит методы для взаимодействия с USB.

13. **Class ConstPCI**

    Класс констант, необходимых для взаимодействия с шиной PCI.

14. **Class Ukmbo_pci**

    Содержит методы для взаимодействия с PCI.

Программная реализация классов представлена в приложениях А-Д.

3.4 Вывод

В этом разделе обосновывается выбор языка программирования и среды разработки, рассмотрена UML-диаграмма основных классов.

Заключение

Во время прохождения практики была достигнута поставленная цель, а именно, была разработана и отлажена библиотека алгоритмов взаимодействия персонального компьютера и контрольной измерительной аппаратурой радиолокационной головкой самонаведения.

В процессе выполнения были решены все задачи: изучены алгоритмы взаимодействия ПК и КИА РГС, протокол канала межблочного обмена, проработаны особенности взаимодействия с PCI, USB и Ethernet. В результате была разработана соответствующая библиотека алгоритмов, которая впоследствии была успешно отлажена.

Список литературы

1. Военный энциклопедический словарь / Пред. Гл. ред. комиссии: С.Ф. Ахромеев. – 2-е изд. – М.: Воениздат, 1986. – 863 с. – 150 000 экз.

2. Куркоткин В.И., Стерлингов В.Л. Самонаведение ракет. – М.: Воениздат, 1963. – 92 с. – (Ракетная техника). – 20 000 экз.

3. Контроллер канала межблочного обмена [Электронный ресурс]. – Режим доступа: https://findpatent.ru/patent/234/2345407.html (дата обращения 01.07.2021).

4. Основные особенности микросхемы FT2232H [Электронный ресурс]. – Режим доступа: http://microsin.net/adminstuff/hardware/ft2232h-dual-uart-fifo-usb-converter.html (дата обращения 05.07.2021)

5. FTDI [Электронный ресурс]. – Режим доступа: https://ftdichip.com/ (дата обращения 05.07.2021)

6. Микросхема mc [Электронный ресурс]. – Режим доступа: http://hardelectronics.ru/mc34063.html (дата обращения 07.07.2021)

7. Документация по C# [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения 02.07.2021)

8. Документация по Visual Studio 2015 [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/visualstudio/vs-2015-archive?view=vs-2019 (дата обращения 07.07.2021)

Приложение А

Приведены листинги основных классов.

Листинг 1 – Основные функции КМБО

```
namespace Kmbo
{
  public class GlobalVar
  {
    public const int KMBO_MAXIMUM_WORDS = 64;
    public static int Type_Device = -1;
  }

  public class TKMBO : TKernel32
  {
    public const UInt32 GENERIC_READ = 0x80000000;
    public const UInt32 GENERIC_WRITE = 0x80000000;
    public const UInt16 OPEN_EXISTING = 3;
    const UInt16 CREATE_NEW = 1;
    public const UInt16 NULL = 0;
    public const UInt16 FILE_SHARE_READ = 0;
    public const UInt16 FILE_SHARE_WRITE = 0;
    public const UInt16 FILE_ATTRIBUTE_NORMAL = 0;
    public const UInt16 FIRST_IOCTL_INDEX = 0x0800;
    public const UInt16 FILE_ANY_ACCESS = 0x0000;
    public const UInt32 FILE_DEVICE_KMBO = 0x00008000;
    public const UInt16 METHOD_BUFFERED = 0;
    const UInt16 METHOD_IN_DIRECT = 1;
    const UInt16 METHOD_OUT_DIRECT = 2;
    const UInt16 METHOD_NEITHER = 3;
    const UInt16 REGIM_REST = 0;
    const UInt16 REGIM_CONTROLLER = 0x1;
  }

  public class TKmboRead
  {
    public UInt16 address;
    public UInt32 count;
    public UInt16 synchro;
    public UInt16[] lpBuffer = new UInt16[GlobalVar.KMBO_MAXIMUM_WORDS];

    public object Clone()
    {
```

```
39        UInt16[] temp = new UInt16[this.lpBuffer.Length];
40        Array.Copy(this.lpBuffer, temp, this.lpBuffer.Length);
41
42        return new TKmboRead
43        {
44          address = this.address,
45          count = this.count,
46          synchro = this.synchro,
47          lpBuffer = temp
48        };
49      }
50    }
51
52    public class TKmboWrite
53    {
54      public UInt16 address;
55      public UInt32 count;
56      public UInt16 synchro;
57      public UInt16[] lpBuffer = new UInt16[GlobalVar.KMBO_MAXIMUM_WORDS];
58
59      public object Clone()
60      {
61        UInt16[] temp = new UInt16[this.lpBuffer.Length];
62        Array.Copy(this.lpBuffer, temp, this.lpBuffer.Length);
63
64        return new TKmboWrite
65        {
66          address = this.address,
67          count = this.count,
68          synchro = this.synchro,
69          lpBuffer = temp
70        };
71      }
72    }
73
74    public class Ukmbo
75    {
76      Ukmbo_mc UkmboMC = new Ukmbo_mc();
77      Ukmbo_usb UkmboUSB = new Ukmbo_usb();
78      Ukmbo_pci UkmboPCI = new Ukmbo_pci();
79      public int KmboOpen(ref IntPtr pHandle)
80      {
81        Errs_mc.Init();
```

```
82
83          if (UkmboMC.MC_Open().Code == 0)
84          {
85            pHandle = (IntPtr)null;
86            GlobalVar.Type_Device = 2;
87          }
88          else
89          {
90            ErrsUSB.Init();
91
92            if (UkmboUSB.USB_Open().Code == 0)
93            GlobalVar.Type_Device = 1;
94            else
95            {
96              ErrsPCI.Init();
97
98              if (UkmboPCI.PCI_Open(ref pHandle).Code == 0)
99              GlobalVar.Type_Device = 0;
100             else
101             GlobalVar.Type_Device = -1;
102           }
103         }
104
105         return GlobalVar.Type_Device;
106       }
107
108       public RObj KmboClose(IntPtr handle)
109       {
110         RObj res = new RObj();
111
112         switch (GlobalVar.Type_Device)
113         {
114           case 0:
115           res.Code = UkmboPCI.PCI_Close(handle).Code;
116           break;
117           case 1:
118           res.Code = UkmboUSB.USB_Close().Code;
119           break;
120           case 2:
121           res.Code = UkmboMC.MC_Close().Code;
122           break;
123         }
124
```

```
125         return res;
126       }
127
128       public RObj KmboWrite(IntPtr handle, ref TKmboWrite lpWrite)
129       {
130
131         RObj res = new RObj();
132
133         switch (GlobalVar.Type_Device)
134         {
135           case 0:
136           res.Code = UkmboPCI.PCI_Write(handle, ref lpWrite).Code;
137           break;
138           case 1:
139           res.Code = UkmboUSB.USB_Write(lpWrite).Code;
140           break;
141           case 2:
142           res.Code = UkmboMC.MC_Write(lpWrite).Code;
143           break;
144         }
145         return res;
146       }
147
148       public RObj KmboRead(IntPtr handle, ref TKmboRead lpRead)
149       {
150         RObj res = new RObj();
151
152         switch (GlobalVar.Type_Device)
153         {
154           case 0:
155           res.Code = UkmboPCI.PCI_Read(handle, ref lpRead).Code;
156           break;
157           case 1:
158           res.Code = UkmboUSB.USB_Read(ref lpRead).Code;
159           break;
160           case 2:
161           res.Code = UkmboMC.MC_Read(lpRead).Code;
162           break;
163         }
164         return res;
165       }
166     }
167 }
```

Приложение Б

Приведены листинги методов, связанных с МС.

Листинг 2 – Основные функции взаимодействия с МС

```
namespace Kmbo
{
  public class Ukmbo_mc
  {
    static UInt64 init_MC_UDP = 0xC0000007L;
    static UInt64 init_MC_Kmbo = 0xC0000007L;

    static int usPort_MKIO = 0x4001;
    static int usPort_KMBO = 0x4002;

    static Socket sSocket_UDP;
    static Socket sSocket_KMBO;

    const int lenStrIpDest = 16;
    static string strIpDest;

    static byte[] NU_MIL_BC = new byte[12] { 0xCD, 0xAB, 0x01, 0x00, 0x01,
        0x00,
      0x01, 0x00, 0x15, 0x00, 0xBA, 0xDC };
    public static long time_on_wr;
    public static long time_on_rd;
    long time_max_wr = 0;
    long time_max_rd = 0;

    public RObj MC_Open()
    {
      RObj res = new RObj();

      Deinit();
      if (Init("192.168.1.71").Code == 0)
      {
        DeinitCom_Kmbo();
        res.Code = (ulong)InitCom_Kmbo("192.168.1.71").Code;
      }
      else
      {
        res.Code = 0xC0000006L;
      }
```

```
38
39        return res;
40      }
41      public RObj MC_Close()
42      {
43        RObj res = new RObj();
44
45        res.Code = Deinit().Code;
46        res.Code += DeinitCom_Kmbo().Code;
47
48        return res;
49      }
50      public RObj MC_Read(TKmboRead argData)
51      {
52        long li_Freq;
53        Stopwatch stopwatch;
54        long li_TimeNow;
55
56        byte[] bufferReceive = new byte[1460];
57        UInt16[] bf_Receive = new UInt16[730];
58
59        int result = 0;
60        UInt16 Number_Frame_KMBO = 0;
61
62        TKmboRead kr = (TKmboRead)argData.Clone();
63
64        EndPoint ipSender;
65        IPEndPoint ipDest;
66
67        try
68        {
69          ipSender = new IPEndPoint(IPAddress.Any, 0);
70        }
71        catch (SocketException ex)
72        {
73          Console.WriteLine("ERROR: " + ex.ToString() + "\n" + ex.Message);
74          return new RObj(0xC0000005L);
75        }
76
77        RObj res = new RObj(0xC0000007L);
78
79        if (init_MC_Kmbo != 0)
80        {
```

```
81          return res;
82        }
83
84        ArrayList al = new ArrayList();
85
86        al.Add((UInt16)0xabcd);
87        al.Add((UInt16)0x0000);
88        al.Add((UInt16)0x0001);
89
90        al.Add((UInt16)(kr.count & 0x00ff));
91        al.Add((UInt16)(((kr.address << 8) & 0xff00) | 0x0080));
92
93        Number_Frame_KMBO++;
94
95        al[2] = (UInt16)Number_Frame_KMBO;
96        al.Add((UInt16)0xdcba);
97
98        try
99        {
100         ipDest = new IPEndPoint(IPAddress.Parse(strIpDest), usPort_KMBO);
101       }
102       catch (SocketException ex)
103       {
104         Console.WriteLine("ERROR: " + ex.ToString() + "\n" + ex.Message);
105         return new RObj(0xC0000005L);
106       }
107
108       sSocket_KMBO.SendTo(StructToByteArray(al), ipDest);
109
110       result = sSocket_KMBO.ReceiveFrom(bufferReceive, ref ipSender);
111
112       if (result > 0)
113       {
114         Buffer.BlockCopy(bufferReceive, 0, bf_Receive, 0, bufferReceive.
              Length);
115
116         if (bf_Receive[0] == 0xabcd)
117         {
118           for (int i = 4; i < bf_Receive.Length / 2; i++)
119           {
120             if ((bf_Receive[i] & 0xff00) == ((kr.address << 8) & 0xff00))
121             {
122               Buffer.BlockCopy(bf_Receive, i + 1, kr.lpBuffer, 0, (int)kr.
```

```
                count);
123
124             kr = argData;
125             res.Code = 0;
126             break;
127           }
128           if (bf_Receive[i] == 0xdcba)
129           {
130             break;
131           }
132         }
133       }
134     }
135
136     return res;
137   }
138   public RObj MC_Write(TKmboWrite argData)
139   {
140     long li_Freq;
141     Stopwatch stopwatch;
142     long li_TimeNow;
143
144     RObj res = new RObj(0xC0000007L);
145
146     byte[] bufferReceive = new byte[1460];
147     UInt16[] bf_Receive = new UInt16[730];
148
149     int result;
150     UInt16 Number_Frame_KMBO = 0;
151
152     TKmboWrite wr = (TKmboWrite)argData.Clone();
153
154     EndPoint ipSender;
155
156     try
157     {
158       ipSender = new IPEndPoint(IPAddress.Any, 0);
159     }
160     catch (SocketException ex)
161     {
162       Console.WriteLine("ERROR: " + ex.ToString() + "\n" + ex.Message);
163       return new RObj(0xC0000005L);
164     }
```

```
165
166         if (init_MC_Kmbo != 0)
167         {
168           return res;
169         }
170
171         ArrayList al = new ArrayList();
172
173         al.Add((UInt16)0xabcd);
174         al.Add((UInt16)0x0000);
175         al.Add((UInt16)0x0001);
176
177         al.Add((UInt16)(wr.count & 0x00ff));
178         al.Add((UInt16)((wr.address << 8) & 0xff00));
179
180         for (int i = 0; i < wr.count; i++) al.Add(wr.lpBuffer[i]);
181         Number_Frame_KMBO++;
182
183         al[2] = (UInt16)Number_Frame_KMBO;
184         al.Add((UInt16)0xdcba);
185
186         IPEndPoint ipDest;
187
188         try
189         {
190           ipDest = new IPEndPoint(IPAddress.Parse(strIpDest), usPort_KMBO);
191         }
192         catch (SocketException ex)
193         {
194           Console.WriteLine("ERROR: " + ex.ToString() + "\n" + ex.Message);
195           return new RObj(0xC0000005L);
196         }
197
198         sSocket_KMBO.SendTo(StructToByteArray(al), ipDest);
199
200         result = sSocket_KMBO.ReceiveFrom(bufferReceive, ref ipSender);
201
202         if (result > 0)
203         {
204           Buffer.BlockCopy(bufferReceive, 0, bf_Receive, 0, bufferReceive.
                Length);
205           if (bf_Receive[0] == 0xabcd) res.Code = 0;
206         }
```

```
207
208       return res;
209     }
210
211     private RObj Init(string argStrIpDest)
212     {
213       if (init_MC_UDP == 0) return new RObj(0);
214
215       strIpDest = argStrIpDest;
216
217       try
218       {
219         sSocket_UDP = new Socket(AddressFamily.InterNetwork, SocketType.
              Dgram, ProtocolType.Udp);
220       }
221       catch (SocketException ex)
222       {
223         Console.WriteLine("ERROR: " + ex.ToString() + "\n" + ex.Message);
224         return new RObj(0xC0000001L);
225       }
226
227       try
228       {
229         sSocket_UDP.ReceiveTimeout = 50;
230       }
231       catch (SocketException ex)
232       {
233         Console.WriteLine("ERROR: " + ex.ToString() + "\n" + ex.Message);
234         return new RObj(0xC0000003L);
235       }
236
237       try
238       {
239         sSocket_UDP.SendTimeout = 50;
240       }
241       catch (SocketException ex)
242       {
243         Console.WriteLine("ERROR: " + ex.ToString() + "\n" + ex.Message);
244         return new RObj(0xC0000004L);
245       }
246
247       Send_NU_UDP();
248       init_MC_UDP = Read_NU_UDP().Code;
```

```
249
250         return new RObj(init_MC_UDP);
251       }
252
253       private RObj Send_NU_UDP()
254       {
255         IPEndPoint ipDest;
256
257         try
258         {
259           ipDest = new IPEndPoint(IPAddress.Parse(strIpDest), usPort_MKIO);
260         }
261         catch (SocketException ex)
262         {
263           Console.WriteLine("ERROR: " + ex.ToString() + "\n" + ex.Message);
264           return new RObj(0xC0000005L);
265         }
266
267         sSocket_UDP.SendTo(NU_MIL_BC, ipDest);
268
269         return new RObj(0);
270       }
271       private RObj Read_NU_UDP()
272       {
273         int result;
274         EndPoint remoteIP;
275
276         try
277         {
278           remoteIP = new IPEndPoint(IPAddress.Any, 0);
279         }
280         catch (SocketException ex)
281         {
282           Console.WriteLine("ERROR: " + ex.ToString() + "\n" + ex.Message);
283           return new RObj(0xC0000005L);
284         }
285
286         byte[] bufferReceive = new byte[1460];
287         StringBuilder str = new StringBuilder();
288
289         result = sSocket_UDP.ReceiveFrom(bufferReceive, ref remoteIP);
290
291         if (result > 0)
```

```
292         {
293           str.Append(Encoding.Unicode.GetString(bufferReceive, 0, result));
294           return new RObj(0);
295         }
296         else return new RObj(0xC0000008L);
297       }
298
299       private RObj Deinit()
300       {
301         if (init_MC_UDP != 0) return new RObj(0xC0000007L);
302
303         sSocket_UDP.Close();
304         init_MC_UDP = 0xC0000007L;
305
306         return new RObj(init_MC_UDP);
307       }
308
309       private RObj InitCom_Kmbo(string argStrIpDest)
310       {
311         RObj result = new RObj();
312         EndPoint ipAddr;
313
314         if (init_MC_Kmbo == 0) return result;
315
316         strIpDest = argStrIpDest;
317
318         try
319         {
320           sSocket_KMBO = new Socket(AddressFamily.InterNetwork, SocketType.
                Dgram, ProtocolType.Udp);
321         }
322         catch (SocketException ex)
323         {
324           Console.WriteLine("ERROR: " + ex.ToString() + "\n" + ex.Message);
325           return new RObj(0xC0000001L);
326         }
327
328         try
329         {
330           ipAddr = new IPEndPoint(IPAddress.Any, 0x5003);
331         }
332         catch (SocketException ex)
333         {
```

```
334           Console.WriteLine("ERROR: " + ex.ToString() + "\n" + ex.Message);
335           return new RObj(0xC0000005L);
336         }
337
338         try
339         {
340           sSocket_KMBO.Bind(ipAddr);
341         }
342         catch (SocketException ex)
343         {
344           Console.WriteLine("ERROR: " + ex.ToString() + "\n" + ex.Message);
345           return new RObj(0xC0000002L);
346         }
347
348         init_MC_Kmbo = 0;
349
350         return result;
351       }
352
353       private RObj DeinitCom_Kmbo()
354       {
355         if (init_MC_Kmbo != 0) return new RObj(0xC0000007L);
356
357         sSocket_KMBO.Close();
358
359         init_MC_Kmbo = 0xC0000007L;
360
361         return new RObj(init_MC_Kmbo);
362       }
363
364       static public byte[] StructToByteArray(ArrayList arr)
365       {
366         object[] obj = (arr.ToArray());
367
368         int sizeInBytes;
369
370         byte[] outByte = new byte[0];
371
372         for (int i = 0; i < obj.Count(); i++)
373         {
374           sizeInBytes = Marshal.SizeOf(obj[i]);
375
376           byte[] outArray = new byte[sizeInBytes];
```

```
377
378             IntPtr ptr = Marshal.AllocHGlobal(sizeInBytes);
379
380             Marshal.StructureToPtr(obj[i], ptr, true);
381             Marshal.Copy(ptr, outArray, 0, sizeInBytes);
382             Marshal.FreeHGlobal(ptr);
383             outByte = outByte.Concat(outArray).ToArray();
384           }
385           return outByte;
386         }
387       }
388 }
```

Приложение В

Приведены листинги методов, связанных с PCI.

Листинг 3 – Основные функции взаимодействия с PCI

```
namespace Kmbo
{
  public class Ukmbo_pci
  {
    public class ConstPCI
    {
      public const int KMBO_SET_REGIM_REST = 0;
      public const int KMBO_SET_REGIM_CONTROLLER = 0x1;
    }
    public struct TKmboSetRegim
    {
      public ulong regim;
    }
    public RObj PCI_Open(ref IntPtr pHandle)
    {
      TKmboSetRegim SetRegim;
      pHandle = TKernel32.CreateFile("\\\\.\\kmbo",
      TKMBO.GENERIC_READ | TKMBO.GENERIC_WRITE,
      TKMBO.FILE_SHARE_READ | TKMBO.FILE_SHARE_WRITE,
      TKMBO.NULL,
      TKMBO.OPEN_EXISTING,
      TKMBO.FILE_ATTRIBUTE_NORMAL,
      TKMBO.NULL);
      if (pHandle == (IntPtr)(-1))
      return new RObj(TKernel32.GetLastError());

      SetRegim.regim = ConstPCI.KMBO_SET_REGIM_CONTROLLER;
      return KmboSetRegim(pHandle, ref SetRegim);
    }
    public RObj PCI_Close(IntPtr handle)
    {
      RObj res = new RObj();

      if (handle != (IntPtr)(-1))
      {
        if (TKernel32.CloseHandle(handle) == false)
        res.Code = TKernel32.GetLastError();
        return res;
```

```
39        }
40        return res;
41      }
42      unsafe public RObj PCI_Write(IntPtr handle, ref TKmboWrite lpWrite)
43      {
44        UInt32 ret = 0;
45        RObj res = new RObj();
46
47        UInt32 lu_SetRegime = (UInt32)((TKMBO.FILE_DEVICE_KMBO << 16) |
48        (TKMBO.FILE_ANY_ACCESS << 14) |
49        ((TKMBO.FIRST_IOCTL_INDEX + 1) << 2) |
50        (TKMBO.METHOD_BUFFERED)); //+1 attention
51
52        if (handle != (IntPtr)(-1))
53        {
54          fixed (UInt16* Arr = &lpWrite.address)
55          {
56            if (!TKernel32.DeviceIoControl(handle,
57            lu_SetRegime,
58            (UInt16*)Arr,
59            (UInt32)Marshal.SizeOf(typeof(TKmboWrite)),
60            null,
61            0,
62            &ret,
63            TKMBO.NULL))
64            res.Code = TKernel32.GetLastError();
65          }
66        }
67        return res;
68      }
69      unsafe public RObj PCI_Read(IntPtr handle, ref TKmboRead lpRead)
70      {
71        UInt32 ret;
72        RObj res = new RObj();
73
74        UInt32 lu_SetRegime = (UInt32)((TKMBO.FILE_DEVICE_KMBO << 16) |
75        (TKMBO.FILE_ANY_ACCESS << 14) |
76        ((TKMBO.FIRST_IOCTL_INDEX) << 2) |
77        (TKMBO.METHOD_BUFFERED));
78        if (handle != (IntPtr)(-1))
79        {
80          fixed (UInt16* Arr = &lpRead.address)
81          {
```

```
82            if (!TKernel32.DeviceIoControl(handle,
83            lu_SetRegime,
84            (UInt16*)Arr,
85            (UInt32)Marshal.SizeOf(typeof(TKmboRead)),
86            (UInt16*)Arr,
87            (UInt32)Marshal.SizeOf(typeof(TKmboRead)),
88            &ret,
89            TKMBO.NULL))
90              res.Code = TKernel32.GetLastError();
91          }
92        }
93        return res;
94      }
95
96      unsafe RObj KmboSetRegim(IntPtr handle, ref TKmboSetRegim lpSetRegim)
97      {
98        UInt32 ret;
99        RObj res = new RObj();
100
101       if (handle != (IntPtr)(-1))
102       {
103         UInt32 lu_SetRegime = (UInt32)((TKMBO.FILE_DEVICE_KMBO << 16) |
104         (TKMBO.FILE_ANY_ACCESS << 14) |
105         ((TKMBO.FIRST_IOCTL_INDEX + 5) << 2) |
106         (TKMBO.METHOD_BUFFERED));
107
108         fixed (TKmboSetRegim* Arr = &lpSetRegim)
109         {
110           if (!TKernel32.DeviceIoControl(handle,
111           lu_SetRegime,
112           (UInt16*)Arr,
113           (UInt32)Marshal.SizeOf(typeof(TKmboSetRegim)),
114           null,
115           0,
116           &ret,
117           TKMBO.NULL))
118             res.Code = TKernel32.GetLastError();
119         }
120       }
121       return res;
122     }
123   }
124 }
```

Приложение Г

Приведены листинги методов, связанных с USB.

Листинг 4 – Основные функции взаимодействия с USB

```
namespace Kmbo
{
  class Ukmbo_usb
  {
    FTDI ftdi = new FTDI();

    public RObj USB_Open()
    {
      FTDI.FT_STATUS ftStatus;

      ftStatus = ftdi.OpenByDescription("USB <-> Serial Cable A");

      if (ftStatus == FTDI.FT_STATUS.FT_OK)
      {
        ftStatus = ftdi.ResetDevice();
        if (ftStatus != FTDI.FT_STATUS.FT_OK) return new RObj((ulong)
          ftStatus);

        ftStatus = ftdi.SetBitMode(0xff, 0x00);
        if (ftStatus != FTDI.FT_STATUS.FT_OK) return new RObj((ulong)
          ftStatus);

        ftStatus = ftdi.SetTimeouts(1, 1);
        if (ftStatus != FTDI.FT_STATUS.FT_OK) return new RObj((ulong)
          ftStatus);

        return new RObj(0);
      }
      else return new RObj(2);
    }

    public RObj USB_Close()
    {
      return new RObj((ulong)ftdi.Close());
    }

    public RObj USB_Write(TKmboWrite lpWrite)
    {
```

```
36          UInt64 res = 0;
37          Int16 Reg = 0;
38          FTDI.FT_STATUS ftStatus;
39
40          UInt16 Adr;
41          UInt16 N;
42          UInt16[] m = new ushort[62];
43          bool s;
44
45          Adr = lpWrite.address;
46          N = (UInt16)lpWrite.count;
47
48          Buffer.BlockCopy(lpWrite.lpBuffer, 0, m, 1, (int)(lpWrite.count *
                sizeof(UInt16)));
49
50          if (lpWrite.synchro == 1) s = true;
51          else s = false;
52
53          UInt32 RxBytes;
54          byte[] TxBuffer = new byte[256];
55          byte[] RxBuffer = new byte[2];
56          UInt32 dwBytesToWrite = 0;
57          UInt32 dwBytesWritten = 0;
58          UInt32 dwBytesReceived = 0;
59
60          if (N > 62) N = 62;
61          m[0] = (UInt16)((Adr << 8) + (N << 1));
62
63          UInt16 n = 0;
64          UInt16 a;
65
66          for (int i = 1; i <= 15; i++)
67          {
68            a = (UInt16)(m[0] >> i);
69            if ((a & 0x0001) == 0x0001) n++;
70          }
71          if (n % 2 == 0) m[0]++;
72
73          dwBytesToWrite = (UInt32)(N + 1) * 2 + 1;
74          Reg |= 1;
75
76          if (s == false) Reg &= ~2;
77          else Reg |= 2;
```

```
78
79          TxBuffer[0] = (byte)Reg;
80          for (int i = 1; i <= ((N + 1) * 2); i++)
81          if (i % 2 == 0)
82          TxBuffer[i] = (byte)(m[(i >> 1) - 1] & 0x00FF);
83          else
84          TxBuffer[i] = (byte)(m[i >> 1] >> 8);
85
86          ftStatus = ftdi.Write(TxBuffer, dwBytesToWrite, ref dwBytesWritten);
87          if (ftStatus != FTDI.FT_STATUS.FT_OK) return new RObj((ulong)
                ftStatus);
88
89          RxBytes = 1;
90          ftStatus = ftdi.Read(RxBuffer, RxBytes, ref dwBytesReceived);
91          if (ftStatus != FTDI.FT_STATUS.FT_OK) return new RObj((ulong)
                ftStatus);
92
93          if (RxBytes != dwBytesReceived) res = 19;
94          if ((RxBuffer[0] & 0xF9) != 0x20) res = 20;
95          if ((RxBuffer[0] & 0x04) == 0x04) res = 21;
96          if ((RxBuffer[0] & 0x02) == 0x02) res = 22;
97
98          return new RObj(res);
99      }
100
101     public RObj USB_Read(ref TKmboRead lpRead)
102     {
103         UInt64 res = 0;
104         Int16 Reg = 0;
105         FTDI.FT_STATUS ftStatus;
106
107         UInt16 Adr;
108         UInt16 N;
109         UInt16[] m = new UInt16[62];
110         bool s;
111
112         Adr = lpRead.address;
113         N = (UInt16)lpRead.count;
114         if (lpRead.synchro == 1) s = true;
115         else s = false;
116
117         UInt32 RxBytes;
118         byte[] TxBuffer = new byte[3];
```

```
119         byte[] RxBuffer = new byte[129];
120         UInt32 dwBytesToWrite = 0;
121         UInt32 dwBytesWritten = 0;
122         UInt32 dwBytesReceived = 0;
123 
124         if (N > 63) N = 0;
125         m[0] = (UInt16)((Adr << 8) + (N << 1) + 0x0080);
126 
127         UInt16 n = 0;
128         UInt16 a;
129 
130         for (int i = 1; i <= 15; i++)
131         {
132           a = (UInt16)(m[0] >> i);
133           if ((a & 0x0001) == 0x0001) n++;
134         }
135         if (n % 2 == 0) m[0]++;
136 
137         if (N == 0) N = 64;
138         dwBytesToWrite = 3;
139         Reg |= 1;
140 
141         if (s == false) Reg &= ~2;
142         else Reg |= 2;
143         TxBuffer[0] = (byte)Reg;
144         TxBuffer[1] = (byte)(m[0] >> 8);
145         TxBuffer[2] = (byte)(m[0] & 0x00FF);
146 
147         ftStatus = ftdi.Write(TxBuffer, dwBytesToWrite, ref dwBytesWritten);
148         if (ftStatus != FTDI.FT_STATUS.FT_OK) return new RObj((ulong)
              ftStatus);
149 
150         RxBytes = (UInt32)(2 * N + 1);
151         ftStatus = ftdi.Read(RxBuffer, RxBytes, ref dwBytesReceived);
152         if (ftStatus != FTDI.FT_STATUS.FT_OK) return new RObj((ulong)
              ftStatus);
153         if (RxBytes != dwBytesReceived) res = 19;
154 
155         for (int i = 1; i <= N; i++)
156         m[i] = (UInt16)((RxBuffer[2 * i - 2] << 8) +(RxBuffer[2 * i - 1] & 0
              x00FF));
157 
158         Buffer.BlockCopy(m, 1, lpRead.lpBuffer, 0, (int)(lpRead.count *
```

```
            sizeof(UInt16)));
159
160         if ((RxBuffer[2 * N] & 0xF9) != 0x20) res = 20;
161         if ((RxBuffer[2 * N] & 0x04) == 0x04) res = 21;
162         if ((RxBuffer[2 * N] & 0x02) == 0x02) res = 22;
163
164         return new RObj(res);
165       }
166     }
167 }
```

Приложение Д

Приведены листинги классов, описывающих ошибки.

Листинг 5 – Ошибки

```
namespace Errs
{
  public class BErr
  {
    public const int RUS = 0;
    public const int ENG = 1;

    public static Dictionary<UInt64, String[]> errs;

    public static String getByCode(UInt64 code, int lang)
    {
      if (errs.ContainsKey(code))
        return "Code = " + code.ToString() + ". " + errs[code][0];
      else
      {
        IntPtr lpMsgBuf = IntPtr.Zero;

        if (TKernel32.FormatMessage(TKernel32.FormatMessageFlags.
           FORMAT_MESSAGE_ALLOCATE_BUFFER |
        TKernel32.FormatMessageFlags.FORMAT_MESSAGE_FROM_SYSTEM |
        TKernel32.FormatMessageFlags.FORMAT_MESSAGE_IGNORE_INSERTS,
        IntPtr.Zero,
        (uint)code,
        0,
        ref lpMsgBuf,
        0,
        IntPtr.Zero) == 0)
          return "Code = " + code.ToString() + ". Not defined mistake";

        return "Code = " + code.ToString() + ". " + Marshal.
           PtrToStringAnsi(lpMsgBuf);
      }
    }

    public static void Add(UInt64 code, String msgEng, String msgRus)
    {
      errs.Add(code, new String[2] { msgEng, msgRus });
    }
```

```
37  }
38
39  public class RObj
40  {
41    private UInt64 _code = 0;
42    private String _msg;
43
44    public UInt64 Code
45    {
46      get
47      {
48        return _code;
49      }
50
51      set
52      {
53        _code = value;
54        _msg = BErr.getByCode(_code, BErr.ENG);
55      }
56    }
57
58    public String Msg
59    {
60      get
61      {
62        return _msg;
63      }
64    }
65
66    public RObj()
67    {
68      _msg = BErr.getByCode(_code, BErr.ENG);
69    }
70
71    public RObj(UInt64 code)
72    {
73      this._code = code;
74      _msg = BErr.getByCode(code, BErr.ENG);
75    }
76
77    public RObj(int lang)
78    {
79      _msg = BErr.getByCode(_code, lang);
```

```
80      }
81
82      public RObj(UInt64 code, int lang)
83      {
84        this._code = code;
85        _msg = BErr.getByCode(code, lang);
86      }
87    }
88
89    public class Errs_mc : BErr
90    {
91      static public void Init()
92      {
93        if (errs != null)
94        errs.Clear();
95
96        errs = new Dictionary<UInt64, String[]>()
97        {
98          [0] = new String[2]{ "OK"},
99
100         [0xC0000001L] = new String[2] { "Creation of socket for
              transfering data was FAILED"},
101
102         [0xC0000002L] = new String[2] { "Binding a socket was FAILED,
              error in calling function 'bind'"},
103
104         [0xC0000003L] = new String[2] { "Error in setting timeout for
              receiving the data from clients to socket"},
105
106         [0xC0000004L] = new String[2] { "Error in setting timeout for
              sending the data to clients from socket"},
107
108         [0xC0000005L] = new String[2] { "Error in creation of connection
              point"},
109
110         [0xC0000006L] = new String[2] { "Connection with mc was FAILED"},
111
112         [0xC0000007L] = new String[2] { "Error in initialization of module
               mc"},
113
114         [0xC0000008L] = new String[2] { "There is no available data"},
115
116       };
```

```
117     }
118   }
119
120     public class ErrsPCI : BErr
121   {
122     static public void Init()
123     {
124       if (errs != null)
125       errs.Clear();
126
127       errs = new Dictionary<UInt64, String[]>()
128       {
129         [0] = new String[2]{ "OK"},
130
131         [0xE0000001L] = new String[2]{ "The zero-length buffer was passed
              to the driver for receiving/transmitting"},
132
133         [0xE0000002L] = new String[2]{ "The buffer with too large length
              was passed to the driver for receiving/transmitting"},
134
135         [0xE0000003L] = new String[2]{ "The buffer with incorrect length
              was passed to the driver for executing command"},
136
137         [0xE0000004L] = new String[2]{ "The incorrect data was passed to
              the driver for executing command"},
138
139         [0xE0000005L] = new String[2]{ "The attribute 'Counter work' was
              set after the exchange"},
140
141         [0xE0000006L] = new String[2]{ "The attribute 'Parity bit control'
               was set after the exchange"},
142
143         [0xE0000007L] = new String[2]{ "No interruption at the end of the
              exchange"}
144       };
145     }
146   }
147
148     public class ErrsUSB : BErr
149   {
150     static public void Init()
151     {
152       if (errs != null)
```

```
153         errs.Clear();
154 
155         errs = new Dictionary<UInt64, String[]>()
156         {
157           [0] = new String[2]{ "OK"},
158 
159           [1] = new String[2]{ "The descriptor is incorrect"},
160 
161           [2] = new String[2]{ "The USB device is not found"},
162 
163           [3] = new String[2]{ "The channel is not initialized"},
164 
165           [4] = new String[2]{ "Input/Output error"},
166 
167           [5] = new String[2]{ "Nothing to transmit"},
168 
169           [6] = new String[2]{ "The resource is unavailable"},
170 
171           [7] = new String[2]{ "Incorrect exchange rate"},
172 
173           [8] = new String[2]{ "Erasing is not available"},
174 
175           [9] = new String[2]{ "Recording is impossible"},
176 
177           [10] = new String[2]{ "Recording error"},
178 
179           [11] = new String[2]{ "EEPROM reading is impossible"},
180 
181           [12] = new String[2]{ "EEPROM recording is impossible"},
182 
183           [13] = new String[2]{ "EEPROM erasing is impossible"},
184 
185           [14] = new String[2]{ "There is no EEPROM"},
186 
187           [15] = new String[2]{ "The EEPROM is not programmable"},
188 
189           [16] = new String[2]{ "The arguments are incorrect"},
190 
191           [17] = new String[2]{ "Not defined mistake"},
192 
193           [18] = new String[2]{ "W/R error"},
194 
195           [19] = new String[2]{ "Status word is not accepted"},
```

```
196
197             [20] = new String[2]{ "Status word error"},
198
199             [21] = new String[2]{ "Counter work"},
200
201             [22] = new String[2]{ "Parity bit error"}
202         };
203     }
204   }
205 }
```