

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ
РАБОТА БАКАЛАВРА**

на тему

**Технологическое программное обеспечение
микроконтроллера платы дисковых
интерфейсов**

Студент

 03.06.2020
(подпись, дата)

Фадеев Евгений Дмитриевич

Руководитель


(подпись, дата)

Коннов Н.Н.

Нормоконтролёр


(подпись, дата)

Кучин А.В.

Работа допущена к защите (протокол зас. кафедры от 03.06.2020 г. № 12)

Заведующий кафедрой


(подпись)

Митрохин М.А.

Работа защищена с отметкой

23.06.2020 № 21)



(протокол заседания ГЭК от

Секретарь ГЭК


(подпись)

Гурин Е.И.

Пенза 2020

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
 Митрохин М.А.
«30» апреля 2020 г.

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

1. Студент Фадеев Евгений Дмитриевич гр. 16BV1 факультета вычислительной техники.
2. Тема работы: «Технологическое программное обеспечение микроконтроллера платы дисковых интерфейсов».

Тема утверждена приказом ПГУ №0335а/о от "30" апреля 2020 г.

3. Руководитель работы к.т.н., профессор каф. ВТ ПГУ Коннов Н.Н.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Необходимо разработать комплекс программ макетирования программного обеспечения микроконтроллера платы дисковых интерфейсов. Комплекс должен состоять из: программы для микроконтроллера и исполняемых на ПЭВМ программы управления прибором и программы приема отладочных сообщений.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметной области, определить язык программирования, разработать протокол взаимодействия, выбрать микроконтроллер, выбрать операционную систему реального времени, выбрать tcp/ip стек для организации работы по сети.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области выбор языка программирования и микроконтроллера	10	30.04.2020 – 04.05.2020	
2.	Разработка протокола взаимодействия между ПЭВМ и микроконтроллером	10	04.05.2020 – 07.05.2020	
3.	Разработка пользовательского интерфейса программы управления	10	07.05.2020 – 14.05.2020	
4.	Разработка программы для микроконтроллера	10	14.05.2020 – 18.05.2020	
5.	Разработка программы приема отладочных сообщений	20	18.05.2020 – 25.05.2020	
6.	Разработка программы управления	20	25.05.2020 – 28.05.2020	
7.	Оформление выпускной работы	20	28.05.2020 – 03.06.2020	

Дата выдачи задания "30" апреля 2020 г.

Руководитель бакалаврской работы
(подпись, дата)



Коннов Н.Н.

Задание к исполнению принял студент
(подпись, дата)

 30.04.2020

Фадеев Е.Д.

Работу к защите допустить
Декан факультета
(подпись, дата)



Фионова Л.Р.

РЕФЕРАТ

Бакалаврская работа содержит 130 листов, 40 рисунков, 12 источников, 3 приложения.

МИКРОКОНТРОЛЛЕР, C, PYTHON, ГОСТ Р 50451-92, ЭМУЛЯЦИЯ, НАКОПИТЕЛЬ, TCP, UDP, ETHERNET.

Цель работы — разработать технологическое системное программное обеспечение, предназначенное для отладки работы со специализированным накопителем.

Системное программное обеспечение состоит из следующих частей: программа управления, расположенное на ПЭВМ, программы приема текстовых отладочных сообщений, расположенной на ПЭВМ и программы для микроконтроллера.

Программа управления предназначена для передачи общих команд взаимодействия с микроконтроллером, а также команд взаимодействия с накопителем, описанных в ГОСТ Р 50451-92, приема ответных сообщений с результатами выполнения команд.

Программа приема отладочных сообщений предназначена для приема сообщений, выдаваемых во время отладки программы для микроконтроллера.

Программа для микроконтроллера предназначена для эмуляции пространства дискового накопителя в ОЗУ микроконтроллера, размеров в 4 Кб, приема команд от программы управления, выполнение команд общего назначения, трансляцию команд по ГОСТ Р 50451-92 на эмулятор пространства дискового накопителя, формирования результатов выполнения команд и отставку их программе управления.

ПГУ 2.090301.37.001 ПЗ

					ПГУ 2.090301.37.001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.	Фадеев Е.Д.				Технологическое программное обеспечение микроконтроллера платы дисковых интерфейсов. Пояснительная записка.	Лит.	Лист
Провер.	Коннов Н.Н.						2
Н. Контр.	Кучин А.В.						1
						ФВТ, каф. ВТ гр. 16ВВ1	
Утверд.	Митрохин М.А.						

Содержание

РЕФЕРАТ	2
Перечень принятых сокращений	7
Введение	8
1. Анализ предметной области	9
1.1. Постановка задачи	9
1.2. Инструменты для реализации ПО для ПЭВМ	10
1.2.1 Инструменты для реализации программы управления прибором	10
1.2.1.1 Анализ требований	10
1.2.1.2 ЯП для реализации программы управления прибором	11
1.2.1.3 PyQt для создания графического интерфейса	11
1.2.1.4 IDE для создания программы управления	12
1.2.2 Инструменты для реализации программы приема отладочных текстовых сообщений	12
1.2.2.3 Анализ требований	12
1.2.2.2 ЯП для реализации программы приема отладочных сообщений	12
1.2.2.3 IDE для создания программы приема отладочных сообщений	13
1.3. Инструменты для реализации ПО для микроконтроллера	13
1.3.1. Анализ требований	13
1.3.2. Микроконтроллер, отладочная плата, программатор	13
1.3.3. OSCPВ	15
1.3.4. Tcp/Ip стек	17
1.3.5. Язык программирования	18
2. Реализация	19
2.1 Описание работы программы управления	19

2.2	Спецификация программы управления.....	20
2.2.1	Модуль Dev_cmd	20
2.2.2	Модуль My_utils	20
2.2.3	Модуль Main	20
2.2.4	Модуль Device.....	23
2.3	Описание работы программы микроконтроллера.....	23
2.3.1	Задачи	23
2.3.2	Спецификация программы для МК.	27
2.3.2.1	Модуль Led	27
2.3.2.2	Модуль Hardware	27
2.3.2.3	Модуль Utils	27
2.3.2.4	Модуль Timer1	28
2.3.2.5	Модуль Systime	28
2.3.2.6	Модуль Log	28
2.3.2.7	Модуль Firmware	29
2.3.2.8	Модуль Call_uip	29
2.3.2.9	Модуль Udp_uip	30
2.3.2.10	Модуль Tcpserver	30
2.3.2.11	Модуль Et.	30
2.3.2.12	Модуль Main.	30
3	Руководство пользователя	31
3.1	Главное окно программы управления.....	31
3.2	Программа приема отладочных текстовых широкове­щательных сообщений.....	33
3.3	Работа без ПЭВМ.....	34
4	Тестирование.....	38
4.1	Команды общего назначения.....	38
4.1.1	Команда «Открыть соединение»	38
4.1.2	Команда «Закрыть соединение»	38
4.1.3	Команда «Считать время МК».....	39

4.1.4	Команда «Проверить соединение»	39
4.1.5	Команда «Чтение состояния МК»	40
4.1.6	Команда «Чтение информации о прошивке»	40
4.1.7	Команда «Назначить IP адресс»	41
4.2	Команды для взаимодействия с накопителем по ГОСТ Р 50451-92	42
4.2.1	Команда «Звонок»	42
4.2.2	Команда «Стереть»	43
4.2.3	Команда «Сброс»	44
4.2.4	Команда «Выдать состояние»	45
4.2.5	Команда «Вывод»	46
4.2.6	Команда «Ввод»	47
4.2.7	Команда «Диагностика»	48
5	Безопасность жизнедеятельности	50
	Заключение	52
	Список использованных источников	53
	Приложение А. Фрагменты листингов программы	54
	Приложение А1. Листинг программы управления.....	54
	Файл «PVM.py»	54
	Файл «dev_cmd.py»	69
	Файл «net_cfg_ini.py»	69
	Файл «dev.py».....	71
	Приложение А2. Листинг программы приема отладочных текстовых сообщений.	74
	Файл «udp_log.py»	74
	Приложение А3. Листинг программы для микроконтроллера.	75
	Файл «et.c»	75
	Файл «et.h»	83
	Файл «format_process.c»	85
	Файл «led.c»	87

Файл «hardware.c»	89
Файл «utils.c»	91
Файл «systime.c»	94
Файл «net.c»	95
Файл «log.c»	97
Файл «firmware.c»	100
Файл «call_uip.c»	102
Файл «udp_uip.c»	105
Файл «controls.c»	107
Файл «main.c»	108
Файл «main.h»	114
Приложение В. Графическая часть	117
Приложение С. Презентация.....	122

Перечень принятых сокращений

МК – микроконтроллер;

ТПО – технологическое программное обеспечение;

ПО – программное обеспечение;

ОСРВ – операционная система реального времени;

ГОСТ – государственный стандарт;

GUI (Graphical user interface) – графический пользовательский интерфейс;

IDE (Integrated Development Environment) – интегрированная среда разработки;

ПЗУ - постоянное запоминающее устройство;

TCP (Transmission Control Protocol) – протокол управления передачей;

IP (Internet Protocol) – межсетевой протокол;

UDP (User Datagram Protocol) – протокол пользовательских дейтаграмм;

ОЗУ – оперативное запоминающее устройство;

ICMP(Internet Control Message Protocol) – протокол межсетевых управляющих сообщений;

ПЭВМ – персональная электронно-вычислительная машина;

ЯП – язык программирования.

					ПГУ 2.090301.37.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

Пользователями информации сейчас являются миллионы людей и с каждым днем это число растет. Информацию, которой оперируют эти пользователи необходимо в определенные моменты сохранять на какие – то накопители. Современный рынок предлагает огромный выбор накопителей для любых нужд – от оптических накопителей с небольшой емкостью пространства для хранения информации до библиотек ленточных накопителей, способных хранить петабайты информации. Однако существуют сферы деятельности, в которых определяющим фактором является не объем накопителя или скорость записи, а долговечность накопителя, защищенность накопителя от механических повреждений, погодных условий. В таких случаях может потребоваться детальная информация о прохождении команд для работы с накопителем, например необходима полная сводка о повреждениях накопителя. Также огромное количество специализированной техники (транспорт и прочее) может поддерживать работу только с определенными накопителями, которые появились десятки лет назад. В таких случаях обычно встает вопрос о сопровождении такой техники, поэтому требуется разработка устройств, поддерживающих работу с такими накопителями. А именно разработка некоторого устройства управления таким накопителем, которое будет способно выполнять запись, чтение и ряд других функций при работе с накопителем.

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		8

1. Анализ предметной области

1.1. Постановка задачи

Требуется разработать комплекс программ макетирования ПО МК платы дисковых интерфейсов, входящей в состав специализированного прибора (далее по тексту «прибор»). Комплекс должен состоять из: программы для МК и исполняемых на ПЭВМ программы управления прибором и программы приема отладочных сообщений.

Программа для МК должна обеспечивать:

- 1) управление вычислительным процессом с использованием ОСРВ;
- 2) поддержку сетевых протоколов Ethernet, IP, TCP, ICMP, UDP;
- 3) поддержку прикладного протокола взаимодействия ПЭВМ с прибором поверх TCP;
- 4) поддержку специализированного протокола доступа к дисковому накопителю;
- 5) эмуляцию на отладочной плате работы платы штатного устройства;
- 6) эмуляцию пространства дискового накопителя в ОЗУ МК;
- 7) выдачу отладочных широковестьельных текстовых сообщений по протоколу UDP.

Программа управления прибором должна обеспечивать:

- 1) выбор, передачу и прием команд управления прибором по прикладному протоколу ПЭВМ – прибор поверх TCP;
- 2) отображения состояния прибора;
- 3) отображение принятых сообщений от прибора.

Программа приема отладочных сообщений должна обеспечивать вывод принимаемых от прибора отладочных текстовых сообщений по протоколу UDP.

1.2. Инструменты для реализации ПО для ПЭВМ

ПО для ПЭВМ состоит из 2 программ: программа управления прибором и программа для приема отладочных текстовых сообщений.

1.2.1 Инструменты для реализации программы управления прибором

Программа для управления прибором представляет собой клиентское приложение, работающее по протоколу ТСР. Инструменты, реализующие клиентское приложение должны позволять создавать интерфейс, обеспечивающий пользователя удобным и простым взаимодействием с ним. Также если приложение создает разработчик встраиваемого ПО для тестирования своего продукта выбранные инструменты для написания приложения должны обеспечивать быстрое и легкое изменение, редактирование и добавление функционала в соответствии с требованиями встраиваемого ПО.

1.2.1.1 Анализ требований

Приложение должно иметь GUI, выполненный в строгом стиле. Цвета интерфейса должны быть приглушенными и комфортными для глаза пользователя. Немаловажным является удобное и грамотное расположение графических элементов. Так же графические элементы должны быть разбиты по логическим группам.

IDE должна поддерживать средства создание, отображения и редактирования графической части приложения. Это обеспечит не только удобство при разработке и будущей поддержке, но и позволит вносить изменения в графическую составляющую приложения человека, не имеющего навыков программирования.

Важной частью также является выбор ЯП, на котором будет написано приложение. Необходимо чтобы он поддерживал модульность, встроенный событийный аппарат, а также позволял создавать сущности высокого уровня. Все это позволит создать приложение со слабой связью графической части и

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		10

основным алгоритмом работы приложения. Выбранный язык должен быть прост в использовании и написание программ на нем не должно занимать много времени.

1.2.1.2 ЯП для реализации программы управления прибором

Учитывая требования, описанные в предыдущем пункте, ЯП для реализации программы управления был выбран Python. Python – высокоуровневый ЯП общего назначения, поддерживает объектно-ориентированное, структурное, функциональное, обобщенное, и аспектно-ориентированное программирование. В нем реализованы такие удобные для разработки ПО архитектурные черты как, динамическая типизация, автоматическая сборка мусора (освобождение памяти, которую занимают неиспользованные и уничтоженные объекты), механизм обработки исключений, поддержка высокоуровневых структур данных и многое другое. Однако Python не лишен и недостатков. К последним можно отнести: низкую производительность, не подходит для работы с памятью на низком уровне, плохая совместимость разных версий языка.

Для создания программы управления использовался Python версии 3.4.

1.2.1.3 PyQt для создания графического интерфейса

Приложение с GUI могут быть созданы на языке Python с использованием Фреймворка PyQt. PyQt – это набор инструментов для создания GUI для языка программирования Python, выполненный в виде расширения Python. Содержит более 350 классов и больше 6000 функций и методов. Одной из самых полезных черт этого Фреймворка является его кроссплатформенность. Он работает на всех платформах, поддерживаемых Qt: Linux и другие Unix-подобные операционные системы, Windows и Mac OS X. PyQt включает в свой состав Qt Designer – дизайнер GUI, инструмент благодаря которому пользователь не знакомый с языком программирования может создавать графические интерфейсы.

1.2.1.4 IDE для создания программы управления

В качестве IDE для создания программы была выбрана среда разработки PyCharm. PyCharm – IDE для ЯП Python, разработанная компанией JetBrains. Является кросс-платформенной средой разработки, совместимой с ОС Windows, Linux, MacOS. PyCharm поддерживает следующие реализации Python: CPython, IronPython, Jython. IDE обладает огромным количеством возможностей в число которых входят: средства для простой навигации в коде, удобный редактор кода с подсветкой синтаксиса, автоотступом и авто-форматированием, автодополнение кода, интегрированное Unit тестирование, постоянно пополняемая коллекция плагинов (пользователь сам может писать плагины для Pycharm, расширяя ее возможности).

1.2.2 Инструменты для реализации программы приема отладочных текстовых сообщений

Программа приема отладочных текстовых сообщений представляет собой серверное приложение для приема широковещательных сообщений по протоколу UDP.

1.2.2.3 Анализ требований

От приложения требуется лишь принятие по протоколу UDP и вывод на экран текстовых отладочных сообщений, поэтому нет необходимости создавать графический интерфейс, а достаточно обойтись текстовым пользовательским интерфейсом. Выбранный ЯП должен быть прост в использовании.

1.2.2.2 ЯП для реализации программы приема отладочных сообщений

В качестве ЯП был выбран Python (п.1.2.1.2)

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		12

1.2.2.3 IDE для создания программы приема отладочных сообщений

В качестве IDE была выбрана среда разработки IDLE. Она поставляется вместе с Python, является кроссплатформенной. Обладает минималистическим интерфейсом, что удобно для написания небольших скриптов на ЯП Python. Выполняет обычные для IDE задачи: редактирование, просмотр, отладка и запуск программ. В редакторе кода используется подсветка синтаксиса.

1.3. Инструменты для реализации ПО для микроконтроллера

Микроконтроллер – это микросхема, предназначенная для управления электронными устройствами. Он сочетает на одном кристалле функции процессора периферийных устройств, содержит ОЗУ и/или ПЗУ.

1.3.1. Анализ требований

Программа для микроконтроллера должна реализовать следующий функционал: индикация состояния устройства, ТСР сервер для приема команд от ПЭВМ. Сервер должен осуществлять следующие действия: обработка входящих запросов от клиентов (в виде команд), выдача результата о выполнении команд. Также необходимо учесть, что время реакции на внешнее событие должно быть минимальным. Интеграция ОСРВ позволит решить ряд проблем, которые могут возникнуть с проектирование и написанием программы для МК.

1.3.2. Микроконтроллер, отладочная плата, программатор

В качестве микроконтроллера был выбран отечественный 32-разрядный микроконтроллер для авиационного применения 1986BE1T от компании АО «Миландр»(см. рисунок 1). Построен на базе RISC ядра с тактовой частотой до 144 МГц. Содержит 48 Кбайт ОЗУ и 128 Кбайт флэш-памяти программ. Что самое важное, в периферию МК входит цифровой интерфейс Ethernet со скоростью передачи 10/100 Мбит.

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		13

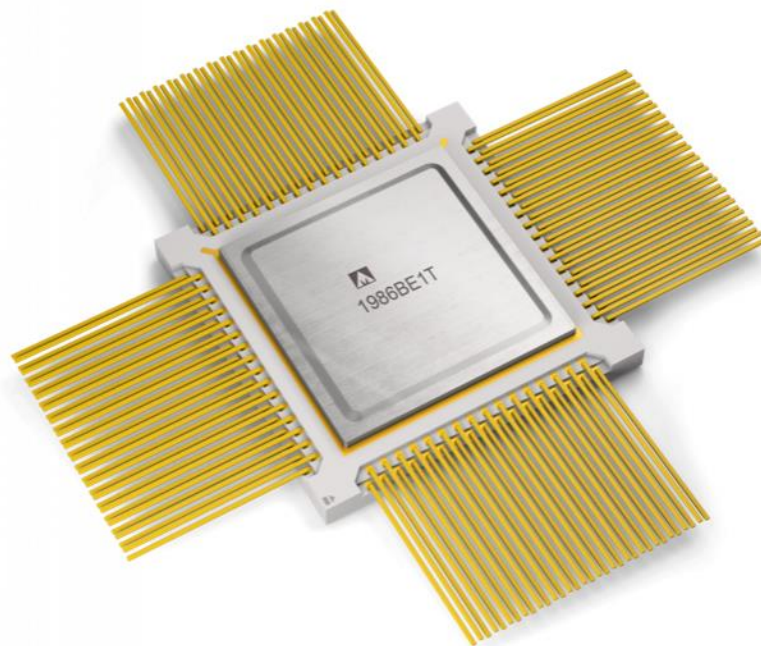


Рисунок 1 – МК 1986BE1T.

МК поставляется вместе с отладочной платой (см. рисунок 2). На ней установлена вся важная для реализации программы для МК периферия (светодиоды, кнопки, разъем для интерфейса Ethernet и т.д.).



Рисунок 2 – отладочная плата для МК 1986BE1T.

В качестве программатора был выбран J-Link Base от компании Segger Microcontroller (см. рисунок 3). J-Link – это JTAG эмулятор с питанием от шины USB, поддерживающий большое количество ядер CPU. Поддержка J-Link интегрирована в большинство IDE для разработки встраиваемого ПО, такие как IAR, Keil и многие другие.



Рисунок 3 – программатор-отладчик J-Link Base.

1.3.3. ОСРВ

Операционная система реального времени — тип операционной системы, основное назначение которой — предоставление необходимого и достаточного набора функций для проектирования, разработки и функционирования систем реального времени на конкретном аппаратном оборудовании. Подразделяются на системы мягкого и жёсткого реального времени. Операционная система, которая может обеспечить требуемое время выполнения задачи реального времени даже в худших случаях, называется операционной системой жёсткого реального времени. Система, которая может обеспечить требуемое время

выполнения задачи реального времени в среднем, называется операционной системой мягкого реального времени. Системы жёсткого реального времени не допускают задержек реакции системы.

В качестве ОСРВ была выбрана FreeRTOS. FreeRTOS – это многозадачная, мультиплатформенная, бесплатная ОС жесткого реального времени с открытым исходным кодом. Большая часть кода системы написана на языке C, вставки на ассемблере применяются лишь там, где невозможно применить C из-за специфики конкретной аппаратной платформы. Поддерживает более 20 архитектур и 50 платформ. FreeRTOS – одна из самых популярных ОСРВ на сегодня.

Основные характеристики:

1. Размер ядра 4 – 9 кбайт.
2. Поддерживается вытесняющая, кооперативная, гибридная многозадачность.
3. Поддерживает сопрограммы и задачи.
4. Развитые средства синхронизации «задача-задача» и «задача-прерывание», включающие: мьютексы, семафоры (двоичные, счетные, рекурсивные), очереди.
5. Возможность отслеживать факт переполнения стека.
6. Нет программных ограничений на количество одновременно выполняемых задач, на количество приоритетов.
7. Нет ограничения на назначения приоритетов.
8. Полностью бесплатна.
9. Поставляется вместе с большим набором примеров и т.д.

1.3.4. Тср/Ір стек

Для взаимодействия с программой управления, реализованной на ПЭВМ необходимо добавить в программу для МК поддержку сети. Для этого был выбран сетевой стек uIP. Он разработан Адамом Дункелсом для встраиваемых устройств с ограниченными ресурсами. Общая структура стека показана на рисунке 4.

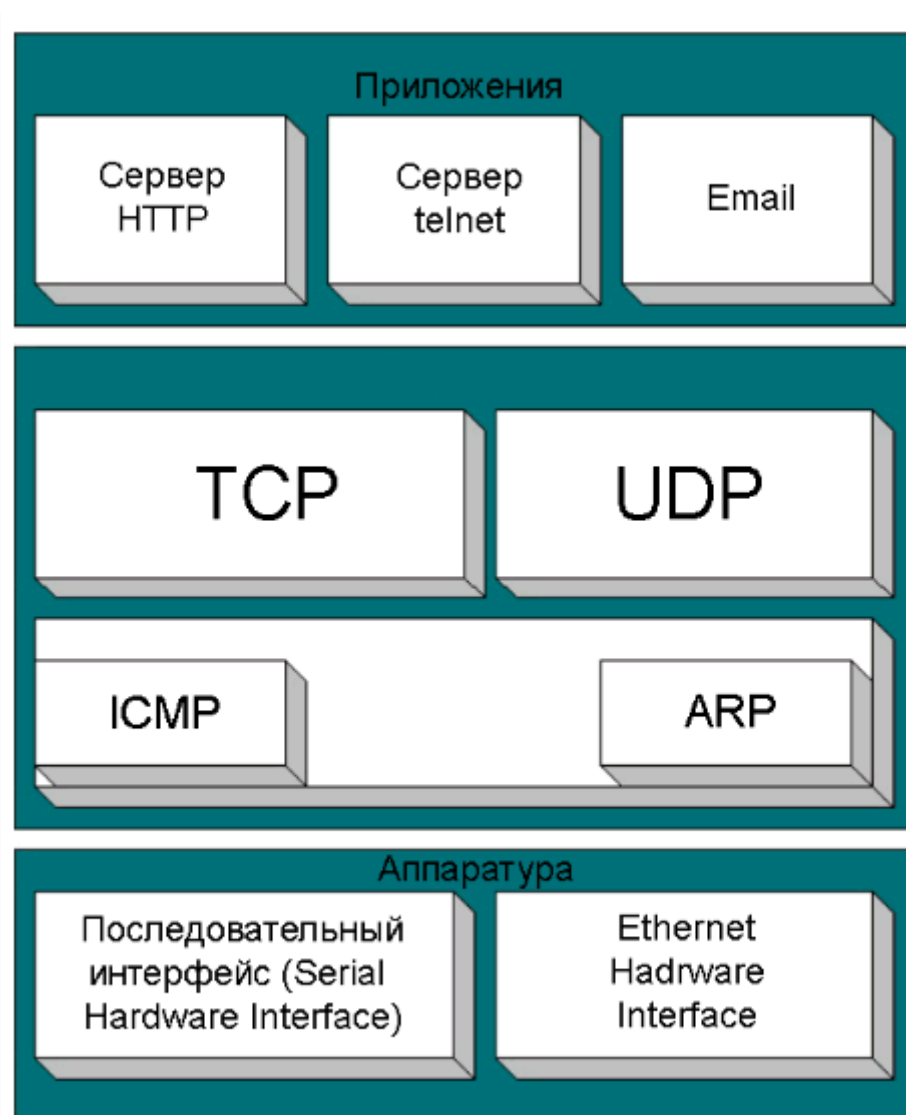


Рисунок 4 – структура стека uIP.

Стек uIP использует один глобальный буфер, размер которого конфигурируется на базе максимального поддерживаемого размера пакета для входящих и исходящих пакетов.

1.3.5. Язык программирования

В качестве языка программирования МК был выбран язык С. С — компилируемый статически типизированный язык программирования общего назначения. Согласно дизайну языка, его конструкции близко сопоставляются типичным машинным инструкциям, благодаря чему он нашёл применение в проектах, для которых был свойственен язык ассемблера, в том числе как в операционных системах, так и в различном прикладном программном обеспечении для множества устройств — от суперкомпьютеров до встраиваемых систем.

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		18

2. Реализация

В данной главе описана реализация программы управления и программы для МК. Освещены основные особенности реализации, связанные с программированием на выбранных языках программирования и средствами разработки. Целью данного раздела является объяснение принципа работы и внутреннее устройство проектируемых программы.

2.1 Описание работы программы управления

Программа для ПЭВМ начинает свою работу, после подключения к серверу, реализованному в МК. Подключаясь к серверу, пользователь обязан указать его ip-адрес и порт, после чего необходимо нажать кнопку подключения «Тср-открыть» после чего производится попытка подключения к нему. В случае успеха или провала в программе для приема широковещательных текстовых отладочных сообщений будет выведен соответственный результат.

После подключения пользователь может начать работу с устройством. Ему доступны следующие команды:

1. Проверка соединения с устройством;
2. Назначение устройству нового IP-адреса, порта и MAC-адреса;
3. Чтение информации о прошивки;
4. Сброс устройства;
5. Чтение времени работы устройства;
6. Команды для работы с накопителем по ГОСТ Р 50451-92
7. Выход из программы управления устройством.

Результаты выполнения команд будут отображаться в окне программы управления устройством, а также в программе для приема широковещательных текстовых сообщений. Для каждой команды определен свой личный, неповторяющийся идентификатор.

2.2 Спецификация программы управления

Приложение разбито на отдельные модули, каждый из которых отвечает за свою функциональность, все они разделены по разным файлам.

2.2.1 Модуль Dev_cmd

В модуле содержатся определения идентификаторов для команд управления МК.

2.2.2 Модуль My_utils

В модуле содержатся полезные функции для работы со строками, временем, коллекциями данных.

В модуле представлены следующие функции:

str_current_time_s() – функция представляющая текущее время в виде строки;

sec2hhmmss() – функция, переводящая секунды в формат часы: минуты: секунды;

bytes2str() – функция, переводящая байты в формат строки;

print_array_byteX() – функция для печати массива байт;

ip_validate() – функция для проверки корректности IP адреса;

delete_delimiters() – функция, которая убирает в строке разделители;

add_to_eigth_nary() – функция, дополняющая строку символами '0' до кратности 8 символов.

net_ip_str_to_ip_tuple() – функция, представляющая строку с ip адресом как кортеж.

2.2.3 Модуль Main

Главный модуль программы управления. Модуль осуществляет установление соединения, формирование, упаковку и отправку команд на серверную часть. В этом модуле представлен один класс, реализующий главное окно программы.

В классе представлены следующие методы:

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		20

`__init__()` – конструктор класса, в нем реализована загрузка формы из файла с расширением `ui`, содержащим форму главного окна приложения, также он инициализирует переменные для хранения IP-адреса и порта, также в нем читается сетевая конфигурация из конфигурационного файла с расширением `.ini`. Устанавливаются переменные для дальнейшей работы значениями по умолчанию.

`init_comboBoxes()` – функция, инициализирующая элемент интерфейса «ComboBox».

`init_ip_and_port ()` – функция, инициализирующая IP – адрес и порт по умолчанию, а именно IP-адрес по умолчанию – 192.168.101.80, а порт по умолчанию – 7139.

`on_pb_ini_cfg_clicked ()` – функция-обработчик нажатия кнопки «Ini». При нажатии появляется окно ввода IP –адреса и порта для подключения к серверу.

`read_ini ()` – функция для чтения данных из конфигурационного файла с расширением «.ini» (IP-адрес, порт).

`set_buttons_tcp_enable ()` – функция, разрешающая действия над кнопкой «tcp открыть».

`set_buttons_tcp_disable ()` – функция, запрещающая действия над кнопкой «tcp открыть».

`send_cmd_and_receive_ack ()` – функция отправляющая упакованную команду и принимающая на нее квитанцию. В параметры передаются код команды, длина данных и сами данные (в случае команд с передачей данных параметры не должны быть равны нулю).

`receive_reply_msg()` – функция принимающая ответное сообщение на команду.

`show_ack ()` – функция, отображающая полученную квитанцию о выполнении команды.

`show_state()` – функция, отображающая состояние прибора и накопителя. Извлекает из тела сообщения соответствующую информацию и выводящую ее в окно программы.

on_pb_read_state_clicked() – функция-обработчик нажатия кнопки «Прочесть состояния прибора». Формирует и отправляет на сервер команду «Прочесть состояние прибора». Получает от него квитанцию. В случае если квитанция равна нулю, получает от сервера сообщение и с помощью функции show_state() разбирает и выводит информацию о приборе на экран.

generate_test_data() – функция генерирует массив байт, указанной в параметрах длины. Используется командой «ВЫВОД» по ГОСТ Р 50451-92.

parse_msg_res_t() – функция, выполняющая разбор ответного входного сообщения на команды по ГОСТ Р 50451-92.

pack_body_vndt() – функция, выполняющая упаковку команды по ГОСТ Р 50451-92.

on_pb_IP_clicked() – функция-обработчик нажатия кнопки «Назначить адрес».

on_pb_vndt_run_clicked() – функция-обработчик нажатия кнопки «Выполнить команду». Упаковывает тело команды по ГОСТ Р 50451-92, отправляет, получает квитанцию, в случае если квитанция равна нулю, принимает ответное сообщение от сервера, разбирает его и выводит на экран данные: код возврата, байт статуса (по ГОСТ Р 50451-92), диагностический байт (по ГОСТ Р 50451-92), длину данных и в случае наличия данных – выводит дамп данных в поле «Протокол» приложения.

on_pb_set_blk_range_t_clicked() – функция-обработчик нажатия кнопки «Задать диапазон блоков».

test_connection() – функция осуществляющая проверку соединения: отправку и получение тестового пакета.

on_pb_check_connection_clicked – функция-обработчик нажатия кнопки «Проверить соединение».

show_fw() – функция, выводящая информацию о прошивке на экран.

on_pb_read_fw_clicked() – функция-обработчик нажатия кнопки «Чтение информации о прошивке».

on_pb_read_time_clicked() – функция-обработчик нажатия кнопки «Чтение времени прибора».

show_time() – функция, выводящая время прибора на экран.

on_pb_reset_clicked() – функция-обработчик нажатия кнопки «Сброс», выполняющая удаленную перезагрузку прибора.

2.2.4 Модуль Device

Модуль Device содержит определения идентификаторов квитанций, которые могут прийти в ответ на отправленную команду. В нем также содержатся функции для работы телом сообщения, которое будет отправлено в устройство или получено от него.

В модуле представлены следующие функции:

extract_header() – функция, которая извлекает заголовок полученного сообщения.

extract_body() – функция, извлекающая тело полученное сообщения.

pack_msg() – функция, собирающая сообщение для отправки.

2.3 Описание работы программы микроконтроллера.

В виду использования ОСРВ программа для микроконтроллера будет представлять из себя систему из быстросменяющих друг друга задач, которые иногда будут взаимодействовать между собой посредством средств межадачного взаимодействия, таких как очереди, счетные/бинарные/рекурсивные семафоры, мьютексы, критические секции. Также реализовано несколько периодичных таймеров, выполняющих определенные действия по истечению определенного промежутка времени и таймеров интервальных, выполняющих однократное действия по истечению определенного времени.

2.3.1 Задачи

Любая программа, выполняющаяся под управлением ОСРВ FreeRTOS, представляет собой множество отдельных независимых задач. Каждая выполняется в своем собственном контексте и только одна задача может

выполняться в один момент времени. Какая именно решает планировщик. Планировщик (Scheduler) – это часть ядра ОСРВ, которая определяет, какая из задач, готовых к выполнению, выполняется в данный конкретный момент времени. Он может приостанавливать, возобновлять выполнение задачи в течение всего жизненного цикла, таким образом создавая вид параллельного выполнения нескольких задач(рисунок 5,6).

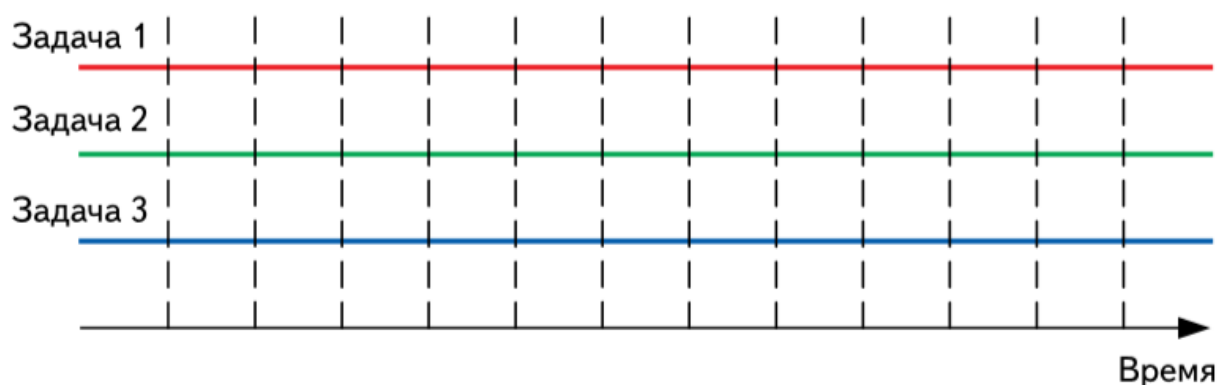


Рисунок 5 – истинно параллельное выполнение задач

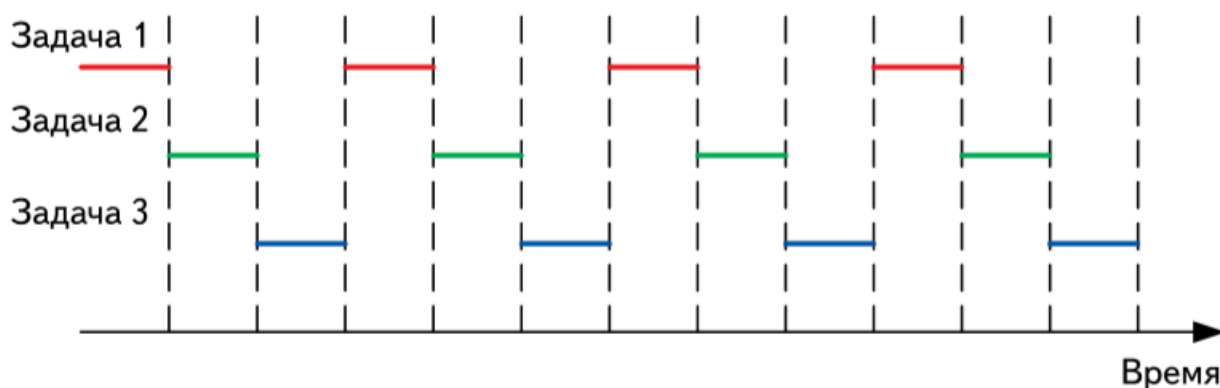


Рисунок 6 – реальное выполнение нескольких задач

Задача может находиться в разных состояниях. Задача выполняется, если в данный момент времени процессор занят ее выполнением. Состояние готовности характеризует задачу, готовую к выполнению, но не выполняющуюся, т.к. процессор занят выполнением другой задачи в данный момент времени. Задача

находится в заблокированном состоянии, если ожидает наступления временного или внешнего события. Приостановленная задача также не получает процессорного времени, однако в отличие от заблокированного состояния, переход в приостановленное состояние и выход из него осуществляется в явном виде с помощью специальных API – функций. Схема жизненного цикла задачи изображена на рисунке 7.

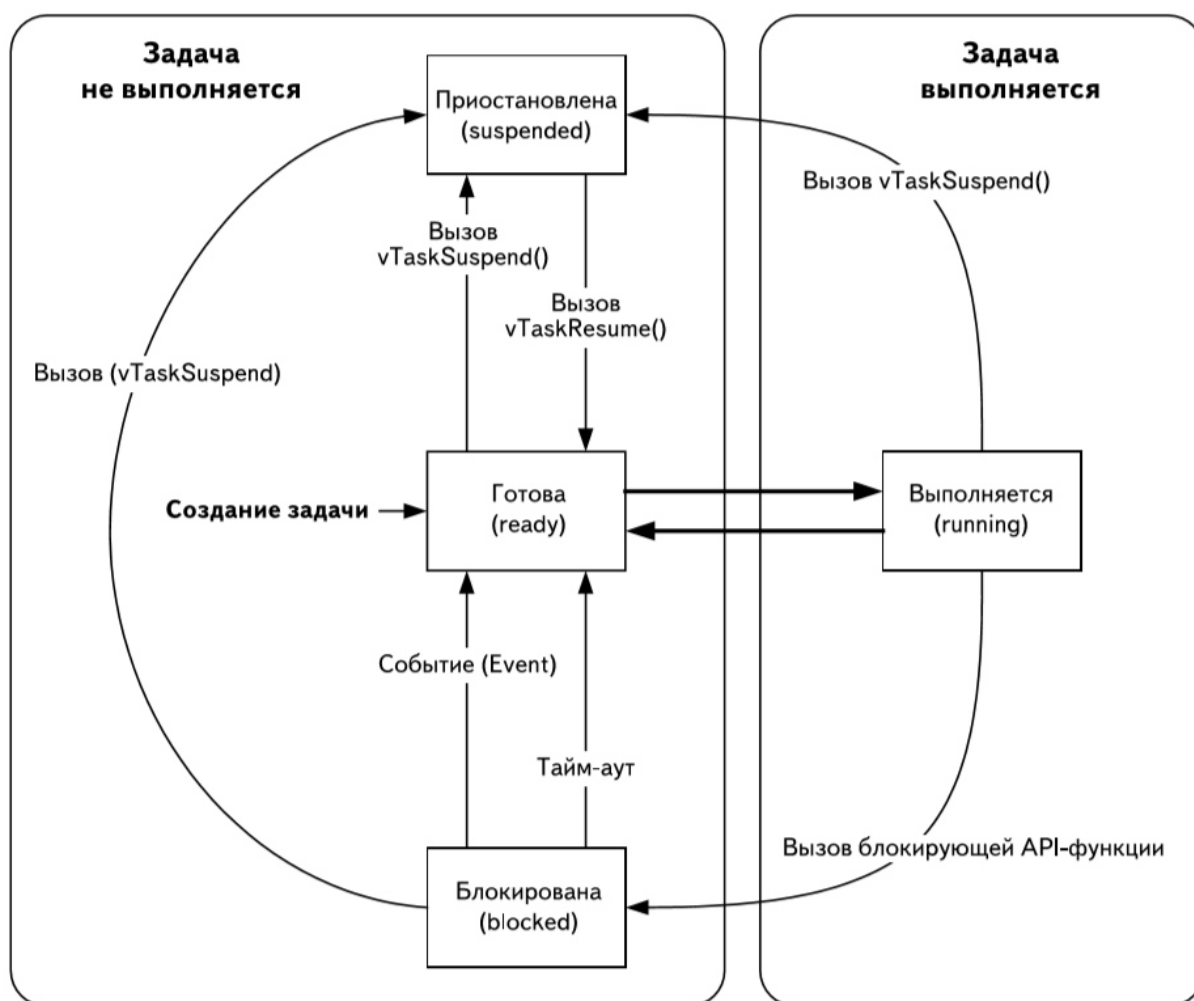


Рисунок 7 – жизненный цикл задачи

В OCPB FreeRTOS реализовано несколько типов многозадачности: вытесняющая, кооперативная и гибридная. Вытесняющая предполагает, что любая выполняющая задача с низким приоритетом прерывается готовой к выполнению задачей с более высоким приоритетом. Переключение между задачами осуществляется через равные кванты времени работы планировщика. Кооперативная многозадачность отличается тем, что планировщик

самостоятельно не может прервать выполнение текущей задачи, каждая задача должна самостоятельно передать управление планировщику. В гибридной многозадачности объединяются концепции вытесняющей и кооперативной многозадачности. В ней вызов планировщика происходит каждый квант времени, однако разработчик может сделать это принудительно в теле задачи. В процессе разработки и написания программы для микроконтроллера была использована гибридная многозадачность.

Задачи в FreeRTOS реализуются в виде C функций. Обязательное требование для такой функции – она не должна ничего возвращать, т.е. иметь тип возвращаемого значения void, и иметь один аргумент типа указатель на void (void*). В листинге 1 представлена структура функции, реализующую задачу.

Листинг 1. Пример функции, реализующую задачу.

```
void vTask1(void* pvParam){
    /* здесь могут быть объявлены здесь, как и в обычной функции*/
    int iVariableExample = 0;
    /* Тело задачи реализовано как бесконечный цикл */
    for(;;){
        iVariableExample++;
        vTaskDelay(1000/portTICK_RATE_MS);
    }
    /* Если все-таки произойдет выход из цикла, задачу необходимо
    уничтожить до выхода из функции */
    vTaskDelete(NULL);
}
```

Основными задачами, которые были реализованы в программе для МК являются:

1. Задача индикации работы устройства
2. Задача сетевого обмена по протоколу TCP с программой управления на ПЭВМ.
3. Задача передачи текстовых отладочных широковещательных сообщений по протоколу UDP программе приема на ПЭВМ.
4. Задача сетевого обмена с ПЭВМ пакетами ICMP.
5. Задача выполнение (эмуляции) команд по интерфейсу ГОСТ Р 50451-92[12].

2.3.2 Спецификация программы для МК.

Программа для МК разбита на отдельные модули, каждый из которых отвечает за свою функциональность, все они разделены по разным файлам.

2.3.2.1 Модуль Led

Модуль Led содержит функции и макросы для работы со светодиодной индикацией работы программы.

Содержит следующие функции:

Led_Init() – функция инициализация портов светодиодов.

Led_series() – функция для мигания определенным светодиодом. В параметры функции передаются пин светодиода, количество миганий, период миганий.

Led_ready() – включение светодиода, который сообщает о готовности МК.

led_READY_toggle() – мигание светодиода, который сообщает о готовности МК.

led_ETH_toggle() – мигание светодиодом, который сообщает об обмене данными с программой управления.

vLed_State_Task() – функция, реализующая задачу индикации работы МК. В ней реализована индикация обмена ICMP пакетами с ПЭВМ, индикация работы по TCP, индикация безошибочной работы МК.

2.3.2.2 Модуль Hardware

Модуль hardware содержит функцию инициализации узлов аппаратуры:

Hardware_init() – функция инициализирует органы управления МК (кнопки), индикаторы (светодиоды), тактирует EEPROM. Также в ней содержится опрос кнопки «Сброс», нажатой при подаче питания.

2.3.2.3 Модуль Utils

Модуль содержит вспомогательный функции для работы со временем и преобразованием представлений IP-адреса и MAC-адреса.

Содержит следующие функции:

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		27

time2string() – функция, которое представляет время в формате день. месяц. год часы: минуты: секунды;

getsElapsedTime() – функция, которая показывает пройденное с момента подачи питания в формате часы: минуты: секунды;

str2mac() – функция, которая преобразует строку в массив, содержащий MAC-адрес;

mac2str() – функция, которая преобразует массив, содержащий MAC в строку;

str2ip() – функция, которая преобразует строки с IP-адресом в IP-адрес;

ip2str() – функция, которая преобразует IP-адрес в строку.

2.3.2.4 Модуль Timer1

Модуль содержит функцию инициализации таймера общего назначения 1.

2.3.2.5 Модуль SysTime

Модуль содержит функции для работы с прошедшим временем:

getElapsedTime() – функция, которая подсчитывает количество прошедшего времени;

sleep() – функция, реализующая задержку с простым МК.

isElapsedTime() – функция, проверяющая, истекло ли время с момента предыдущей проверки.

2.3.2.6 Модуль Log

Модуль реализующий ввод отладочной текстовой информации и ее отправку по протоколу UDP программе приема широковещательных текстовых отладочных сообщений, реализованной на ПЭВМ.

Модуль содержит следующие функции:

Init_log_fifo() – функция инициализации очереди FreeRTOS для хранения массивов с отладочными тестовыми сообщениями;

Prefix() – функция, определяющая префикс, который должен быть присоединен к каждому отладочному сообщению (формат - #время#счетчик отладочных сообщений);

Vlog_put() – функция, присоединяющая префикс к основному отладочному тестовому сообщению;

Log_put() – функция для формирования текстового отладочного сообщения (по аналогии функции printf() из стандартной библиотеки языка C) и помещение его в очередь FreeRTOS для работы с отладочными текстовыми сообщениями;

vLogFlushTask – функция, реализующая задачу FreeRTOS, выводит содержимое очереди в массив с ее очисткой и отправляет массив по протоколу UDP программе для приема текстовых отладочных сообщений, находящейся на ПЭВМ;

log_dump() – функция, помещаю дамп памяти в очередь FreeRTOS для текстовых отладочных сообщений.

2.3.2.7 Модуль Firmware

Модуль предназначен для работы с описаниями прошивки. В нем объявлена структура firmware_info_TypeDef с полями id (идентификатор прошивки), crc (контрольная сумма прошивки), timestamp (временная метка прошивки) и size (размер прошивки).

Модуль содержит следующие функции:

firmware_info_set() – функция, заполняющая поля структуры firmware_info_TypeDef;

getsFirmware() – функция, считывающая поля структуры firmware_info_TypeDef в массив.

2.3.2.8 Модуль Call_uip

Модуль предназначен для инициализации библиотеки uip. Содержит следующие функции: eth_init() – в этой функции инициализируется uIP TCP/IP стек, устанавливается MAC-адрес и инициализируется TCP сервер.

2.3.2.9 Модуль Udp_uip

В модуле представлены функции для функционирования протокола UDP.

2.3.2.10 Модуль Tcpserver

В модуле представлены функции для работы с tcp сервером.(отправка данных, прием, инициализация tcp сервера).

2.3.2.11 Модуль Et.

Модуль содержит реализацию команд интерфейса ГОСТ Р 50451-92 (их эмуляция).

2.3.2.12 Модуль Main.

Главные модуль программы для МК. Содержит следующие задачи:

- 1) Задача индикации работы устройства, с помощью светодиодов;
- 2) Задачу сетевого обмена с ПЭВМ;
- 3) Задачу работы по интерфейсу ГОСТ Р 50451-92;
- 4) Задачу обмена по протоколу ICMP с ПЭВМ.

3 Руководство пользователя

3.1 Главное окно программы управления

После запуска приложения перед пользователем появляется главное окно (рисунок 8).

Программа управления

Начало/конец работы

присоединиться

отключиться

Выход

Адрес прибора
Связь с прибором

IP адрес 192.168.101.81

порт 7137

назначить в данной сессии

Команды протокола обмена

Т код команды 0x8

Команда Т вывод

диапазон блоков накопителя

0 нач. номер бло 0

0 кол-во блоков

Задать

длина данных

гс ?

б.сост. ?

б.диаг. ?

Общие

макс.номер бло ?

Чт.состояния ?

Готов ?

Т ?

Занято ?

Назначить IP адрес 7139 192.168.101.81

Проверить соединение ? ?

Чт. инфо о прошивке

Сброс Время

результат выполнения команды ?

Квитанция ?

Протокол

Рисунок 8 – главное окно программы управления

Главное окно программы имеет следующие основные части:

- группа «Протокол»;
- группа «Команды»;
- группа «Команды протокола обмена»;
- группа «Адрес прибора»;
- группа «Начало/конец работы».

Группа «Протокл» включает в себя 1 поле – поле для вывода результата выполнения команд интерфейса ГОСТ Р 50451-92.

Группа «Команды» включает в себя 1 поле – поле для вывода результата общих команд управления.

Группа «Команды протокола обмена» включает в себя следующие подгруппы:

- подгруппа «Т»;
- подгруппа «Общие»;

В подгруппе «Т» содержатся виджеты для работы с командами по ГОСТ Р 50451-92.

В подгруппе «Общие» содержатся виджеты для работы с общими командами для взаимодействия с МК.

Подгруппа «Т» включает следующие элементы

- кнопка «Команда Т» - при нажатии отправляет команду, выбранную из окна выбора команды, стоящего рядом с кнопкой;
- кнопка «Зачать» - при нажатии задается начальный номер блока и количество блоков накопителя, которые задаются выше в группе «Диапазон блоков» (необходимо для команд «ВЫВОД»[12] и «ВВОД»[12])
- виджет выбор длины данных каждого блока;
- метки, оповещающие о коде возврата, байте состояния[12], диагностическом байте[12].

Подгруппа «Общие» включает следующие элементы:

- Кнопка «Чтение состояние» - при нажатии отправляется команда чтения состояния МК, при успешном прохождении команды заполняются поле «макс.номер блока», метка «Готов», метка «Т», метка «Занято».

- кнопка «Назначить IP адрес» - при нажатии отправляется команда смены сетевого адреса МК, перед нажатием необходимо заполнить рядом стоящие поля порта, IP-адреса и MAC-адреса;
- Кнопка «Проверить соединение» - при нажатии отправляется команда проверки соединения с МК, в случае успеха рядом выводится соответствующая метка;
- Кнопка «Чт. Информации о прошивке» - при нажатии отправляется команда считывающая последнюю прошивку МК, при успешном выполнении команды, информация о прошивке выводится в поле «Команды»;
- Кнопка «Сброс» - при нажатии отправляется команда перезагружающая МК;
- Кнопка «Время» - при нажатии отправляется команда, считывающая время работы МК, в случае успешного прохождения команды время работы выводится в окно «Команды».

Метка «результат выполнения команды» сообщает об успешное выполнение команды.

В группе «Адрес прибора» включает 2 поля: поле для ввода IP-адреса и поле для ввода порта сервера, к которому необходимо присоединиться. Кнопка «Назначить в данной сессии» устанавливает данные настройки на время работы программы управления.

В группе «Начало/конец работы» представлены кнопки для присоединения и отключения от сервера. Также присутствует кнопка правильного выхода из программы «Выход».

3.2 Программа приема отладочных текстовых широковещательных сообщений

В случае если пользователю нужно получить полную информацию о прохождении команды или информацию о работы МК ему необходимо

запустить программу приема отладочных текстовых широковещательных сообщений (рисунок 9).

```

*Python 3.7.4 Shell*
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\Diplom_v.1\udp_log7144.py =====
udp_log7144.py
регистрация пакетов УЗЧД (порт 7144 )
20.06.20 20:35:53.537
|

```

Рисунок 10 – вид программы приема отладочных сообщений (вызов из IDLE)

В случае если необходимо сменить порт прослушивания это необходимо сделать из кода программы в месте *port = <ваше_значение_порта>*.

3.3 Работа без ПЭВМ

В случае если нет доступа к ПЭВМ, можно следить за автономной работой МК с помощью светодиодов, а также выполнять такие функции как сброс устройства и форматирование накопителя (эмуляция) с помощью кнопок. На рисунке 11 представлена индикация работы МК.

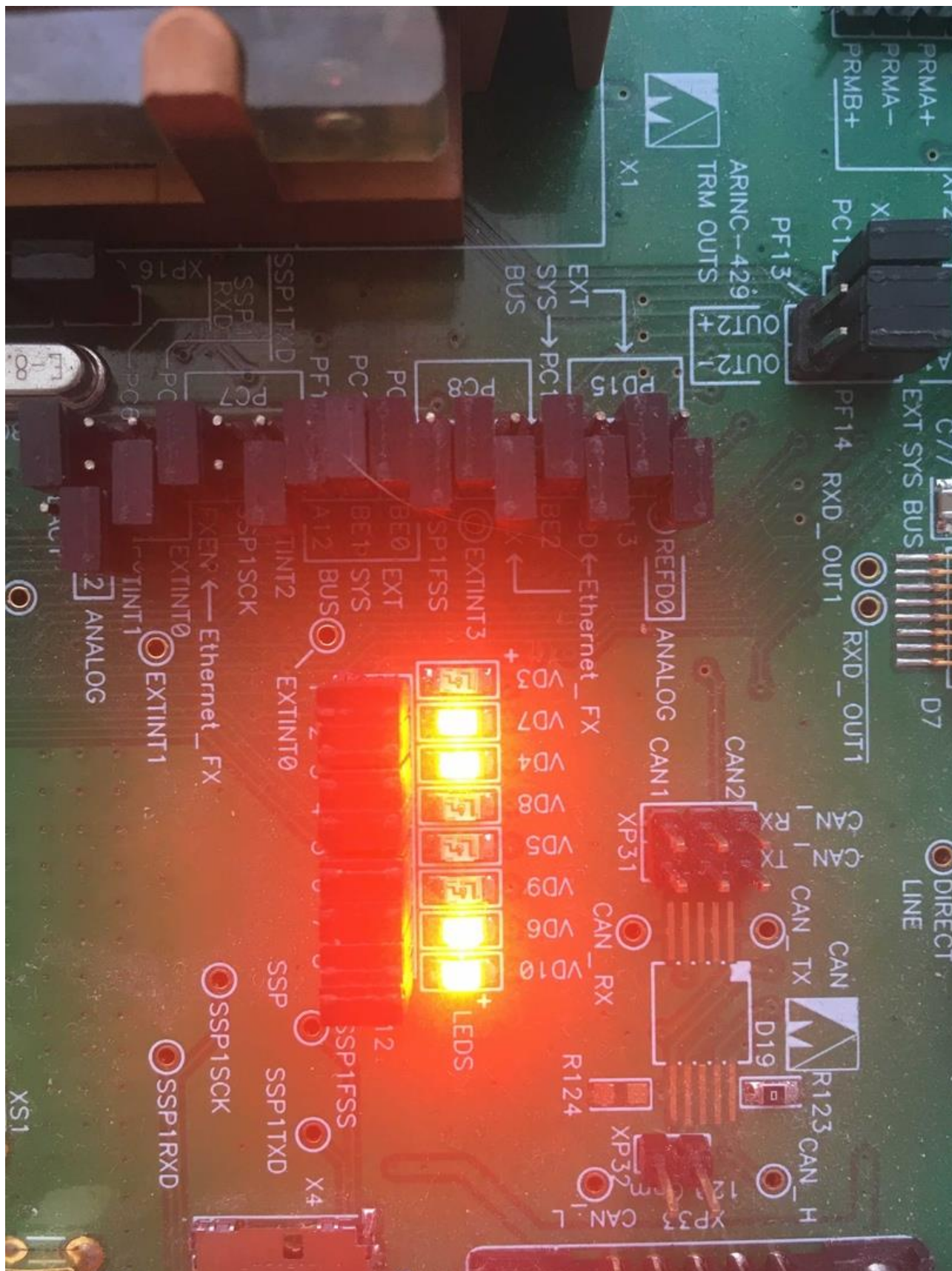


Рисунок 11 – индикация работы МК.

Далее сверху вниз (VD3 – VD10): VD3 показывает безошибочную работу МК (при отсутствии ошибок период мигания – 1 секунда), VD7 показывает что МК

готов к работе (горит постоянно), VD4,VD8,VD5 – зарезервировано, VD9 – показывает факт открытия/закрытия tcp сессии. VD6 – показывает, что МК находится в формате эмуляции команд по ГОСТ Р 50451-92[12] (горит постоянно), VD10 – показывает поступление Ethernet пакета (мигает в случае поступления). Также если у оператора нет доступа к ПЭВМ, он может выполнить команды «Сброс» (перезагрузка МК) и «Форматирование накопителя» (эмуляция форматирования), путем нажатия кнопок (рисунок 12).

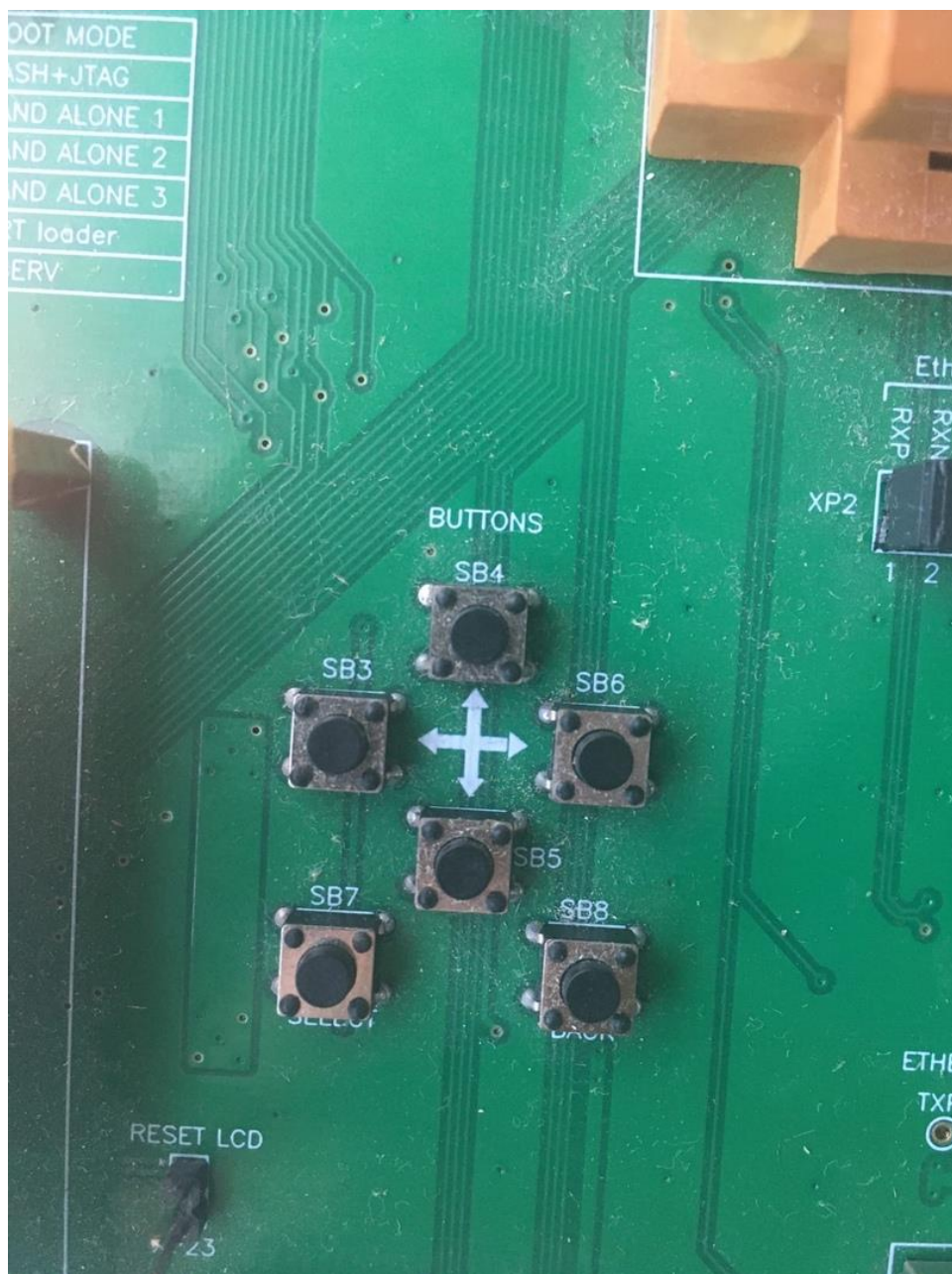


Рисунок 12 – кнопки управления МК.

При нажатии на кнопку «SB3» будет выполнена команда «Сброс» (перезагрузка МК), при нажатии на кнопку «SB5» будет выполнена команда «Форматирование накопителя» (эмитация).

					ПГУ 2.090301.37.001 ПЗ	Лист
						37
Изм.	Лист	№ докум.	Подпись	Дата		

4 Тестирование

В этой главе будут описаны тесты общих команд и команд взаимодействия МК с накопителем по ГОСТ Р 50451-92[12].

4.1 Команды общего назначения

К командам общего назначения относятся следующие команды: «Открыть соединение», «Закрыть соединение», «Считать время МК», «Проверить соединение», «Чтение состояния МК», «Назначить IP адресс», «Чтение информации о прошивке».

4.1.1 Команда «Открыть соединение»

Команда «Открыть соединение» выполняет открытие tcp сессии работы МК с ПЭВМ. Результат работы представлен на рисунке 13.

```
TCP SERVER 1: connection is established  
< dev_form_msg 0x01 msg_body_len=24
```

Рисунок 13 – отладочная информация о прохождении команды «Открыть соединение»

4.1.2 Команда «Закрыть соединение»

Команда «Закрыть соединение» выполняет закрытие tcp сессии работы МК с ПЭВМ. Результат работы представлен на рисунке 14.

```
21.06.20 05:39:33.760  
tcpserver1_appcall closed  
tcpserver_closed  
TCP SERVER 1: connection is closed
```

Рисунок 14 – отладочная информация о прохождении команды «Закрыть соединение».

4.1.3 Команда «Считать время МК»

Команда «Считать время МК» выполняет считывание времени реальной работы МК. Результат работы представлен на рисунках 15,16.

```
21.06.20 05:42:43.877
> tcp_get len=4 opcode=0x12 , seq=2 bodylen=0
< ack_send code=0x0
< dev_form_msg 0xAA msg_body_len=4
21.06.20 05:42:44.267
uptime=00:05:26 (326) s
< dev_form_msg 0x12 msg_body_len=4
```

Рисунок 15 – отладочная информация о прохождении команды «Считать время МК».

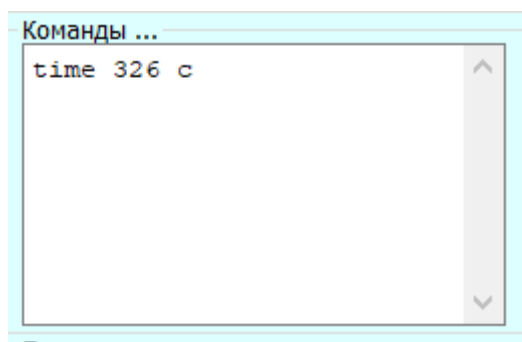


Рисунок 16 –результат работы команды «Считать время МК».

4.1.4 Команда «Проверить соединение»

Команда «Проверить соединение» проверяет есть ли сетевая связь между прибором и ПЭВМ. Результат работы представлен на рисунках 17, 18.

```
21.06.20 05:46:25.020
> tcp_get len=40 opcode=0x2 , seq=2 bodylen=36
< ack_send code=0x0
< dev_form_msg 0xAA msg_body_len=4
21.06.20 05:46:25.489
< msg_test_connection, msg body len=36
< dev_form_msg 0x02 msg_body_len=36
```

Рисунок 17 – отладочная информация о прохождении команды «Проверить соединение».

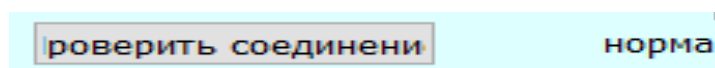


Рисунок 18 –результаты работы команды «Проверить соединение».

4.1.5 Команда «Чтение состояния МК»

Команда «Чтение состояния МК» проверяет готов ли к работе прибор.

Результат работы представлен на рисунках 19, 20.

```
21.06.20 05:52:31.696      -
> tcp_get len=4 opcode=0x1 , seq=1 bodylen=0
< ack_send code=0x0
< dev_form_msg 0xAA msg_body_len=4
21.06.20 05:52:31.856
< dev_form_msg 0x01 msg_body_len=24
```

Рисунок 19 – отладочная информация о прохождении команды «Чтение состояния МК»

Готов 1

Рисунок 20 –результаты работы команды «Чтение состояния МК»

4.1.6 Команда «Чтение информации о прошивке».

Команда «Чтение информации о прошивке» считывает информацию о последней прошивке МК(идентификатор, временную метку, размер). Результат работы представлен на рисунках 21, 22.

```
> tcp_get len=4 opcode=0x11 , seq=0 bodylen=0
< ack_send code=0x0
< dev_form_msg 0xAA msg_body_len=4
21.06.20 06:09:05.030
< dev_form_msg 0x11 msg_body_len=16
```

Рисунок 21 – отладочная информация о прохождении команды «Чтение состояния МК»

Команды ...

```
21.06.20 06:09:05
информация о прошивке:
id      =0x33303030
timestamp =21.06.20
06:01:12
size    =56400
crc     =0xF383B878
```

Рисунок 22 –результаты работы команды «Чтение информации о прошивке».

4.1.7 Команда «Назначить IP адрес».

Команда «Назначить IP адрес» назначает новый сетевой адрес МК (порт, IP-адрес, MAC-адрес). Результат работы представлен на рисунках 23-25.

```
C:\Users\merce>ping 192.168.101.91

Обмен пакетами с 192.168.101.91 по с 32 байтами данных:
Ответ от 192.168.101.91: число байт=32 время<1мс TTL=128
Ответ от 192.168.101.91: число байт=32 время<1мс TTL=128
Ответ от 192.168.101.91: число байт=32 время<1мс TTL=128
Ответ от 192.168.101.91: число байт=32 время=1мс TTL=128

Статистика Ping для 192.168.101.91:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0
    (0% потерь)
Приблизительное время приема-передачи в мс:
    Минимальное = 0мсек, Максимальное = 1 мсек, Среднее = 0 мсек
```

Рисунок 23 – обращение по старому IP-адресу к МК.

```
> tcp_get len=16 opcode=0x20 , seq=0 bodylen=12
CMD_SET_IP_ADDR ip=192.168.101.92 mac=02:02:03:04:05:60 port=7139
cfg_save done
< ack_send code=0x0
< dev_form_msg 0xAA msg_body_len=4
```

Рисунок 24 – отладочная информация о прохождении команды.

```
C:\Users\merce>ping 192.168.101.91

Обмен пакетами с 192.168.101.91 по с 32 байтами данных:
Ответ от 192.168.101.81: Заданный узел недоступен.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.

Статистика Ping для 192.168.101.91:
    Пакетов: отправлено = 4, получено = 1, потеряно = 3
    (75% потерь)

C:\Users\merce>ping 192.168.101.92

Обмен пакетами с 192.168.101.92 по с 32 байтами данных:
Ответ от 192.168.101.92: число байт=32 время<1мс TTL=128
Ответ от 192.168.101.92: число байт=32 время=1мс TTL=128
Ответ от 192.168.101.92: число байт=32 время<1мс TTL=128
Ответ от 192.168.101.92: число байт=32 время<1мс TTL=128

Статистика Ping для 192.168.101.92:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0
    (0% потерь)
Приблизительное время приема-передачи в мс:
    Минимальное = 0мсек, Максимальное = 1 мсек, Среднее = 0 мсек
```

Рисунок 25 –обращение по новому адресу к МК.

4.2 Команды для взаимодействия с накопителем по ГОСТ Р 50451-92

К командам взаимодействия с накопителем относятся команды «Вывод», «Ввод», «Звонок», «Стереть», «Сброс», «Выдать состояние», «Диагностика».

4.2.1 Команда «Звонок»

Команда описана в [12]. В проекте реализована эмуляция этой команды. Команда «Звонок» выдает байт состояния «Готов»[12]. На рисунках 26, 27 представлен результат выполнения команды.

```
21.06.20 17:18:43.361 -- --
> tcp_get len=12 opcode=0x6 , seq=0 bodylen=8
< ack_send code=0x0
< dev_form_msg 0xAA msg_body_len=4
21.06.20 17:18:43.940
  CMD_T_RUN msg_body:
[20000a70,20000af0] sz=128
00000018 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Start Process
Drive_T_Start cmd=0x18
StartTCommand cmd=0x18
CMD_T_RING
et_ResetEraseRing: status byte=0x1
Drive_T_Start retcode8=0x00 stat_byte=0x01 diag_byte=0x00
Finish Process
Drive_T_Termination rc=0x00000000
< msg_send_res rc=0x00000100, data len=0
< msg_send_res opcode=0x05 body_len=1032
[20000650,200006d0] sz=128
00080005 00000100 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Рисунок 26 – вывод отладочной информации по команде «Звонок».

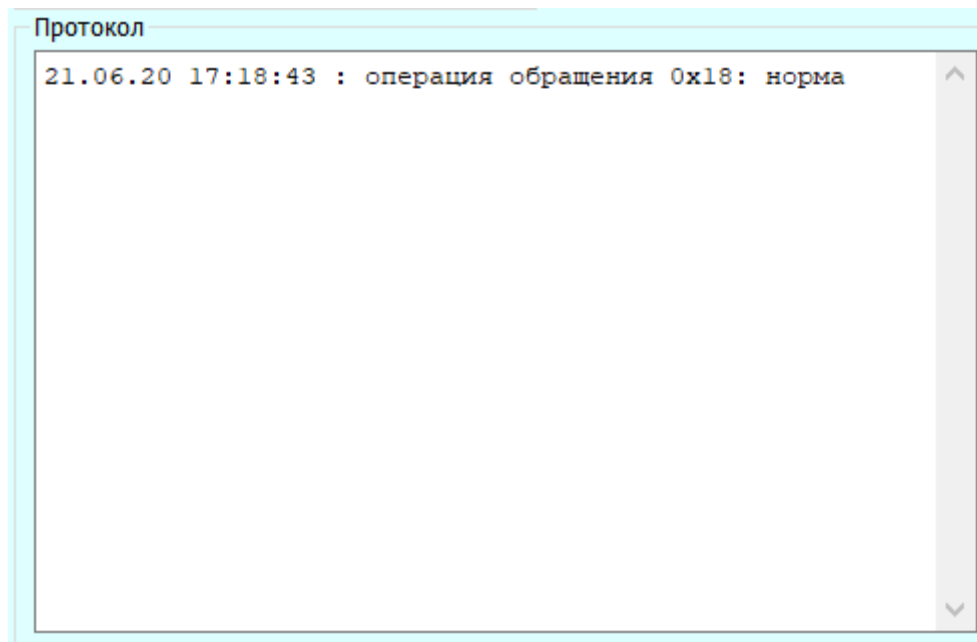


Рисунок 27 – результат выполнения команды «Звонок».

4.2.2 Команда «Стереть»

Команда описана в [12]. В проекте реализована эмуляция этой команды. Команда «Стереть» выдает байт состояния «Готов»[12]. На рисунках 28, 29 представлен результат выполнения команды.

```

21.06.20 17:20:32.792
> tcp_get len=12 opcode=0x6 , seq=1 bodylen=8
< ack_send code=0x0
< dev_form_msg 0xAA msg_body_len=4
21.06.20 17:20:33.050
  CMD_T_RUN msg_body:
[20000a70,20000af0] sz=128
000000D8 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Start Process
Drive_T_Start cmd=0xD8
StartTCommand cmd=0xD8
CMD_T_ERASE
et_ResetEraseRing: status byte=0x1
Drive_T_Start retcode8=0x00 stat_byte=0x01 diag_byte=0x00
Finish Process
Drive_T_Termination rc=0x00000000
< msg_send_res rc=0x00000100, data len=0
< msg_send_res opcode=0x05 body_len=1032
[20000650,200006d0] sz=128
00080105 00000100 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

Рисунок 28 – вывод отладочной информации по команде «Стереть».

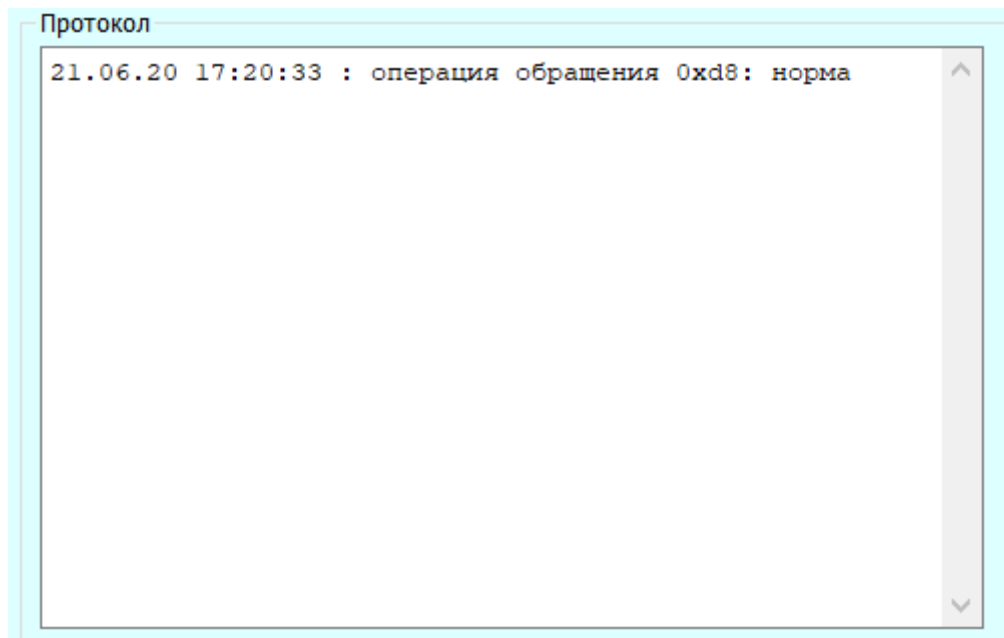


Рисунок 29 – результат работы команды «Стереть».

4.2.3 Команда «Сброс»

Команда описана в [12]. В проекте реализована эмуляция этой команды. Команда «Сброс» выдает байт состояния «Готов»[12]. На рисунках 30, 31 представлен результат выполнения команды.

```
21.06.20 17:21:57.222
> tcp_get len=12 opcode=0x6 , seq=2 bodylen=8
< ack_send code=0x0
< dev_form_msg 0xAA msg_body_len=4
21.06.20 17:21:58.136
  CMD_T_RUN msg_body:
[20000a70,20000af0] sz=128
  000000F8 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Start Process
Drive_T_Start cmd=0xF8
StartTCommand cmd=0xf8
CMD_T_RESET
et_ResetEraseRing: status byte=0x1
Drive_T_Start retcode8=0x00 stat_byte=0x01 diag_byte=0x00
Finish Process
Drive_T_Termination rc=0x00000000
< msg_send_res rc=0x00000100, data len=0
< msg_send_res opcode=0x05 body_len=1032
[20000650,200006d0] sz=128
  00080205 00000100 00000000 00000000 00000000 00000000 00000000 00000000
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Рисунок 30 – вывод отладочной информации по команде «Сброс».

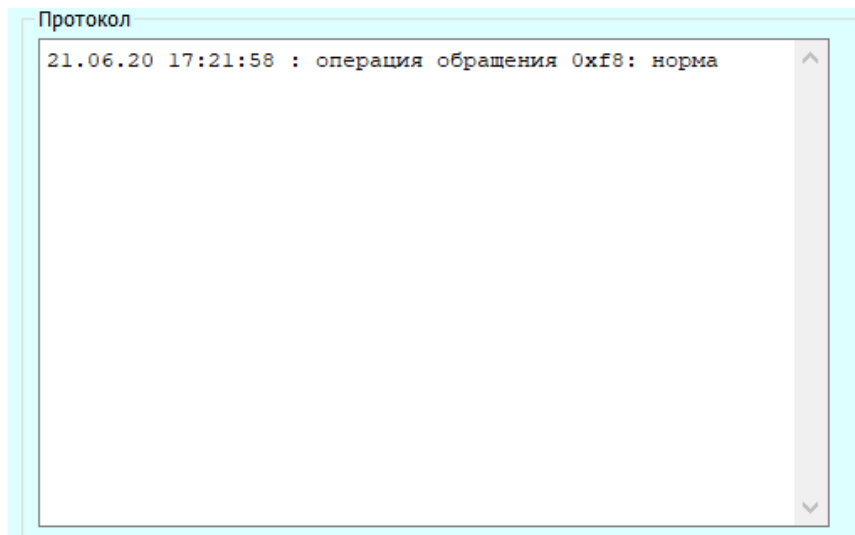


Рисунок 31 – результат выполнения команды «Сброс».

4.2.4 Команда «Выдать состояние»

Команда описана в [12]. В проекте реализована эмуляция этой команды. Команда «Выдать состояние» выдает последний сформированный байт состояния. На рисунках 32, 33 представлен результат выполнения команды.

```

21.06.20 17:24:10.924
> tcp_get len=12 opcode=0x6 , seq=3 bodylen=8
< ack_send code=0x0
< dev_form_msg 0xAA msg_body_len=4
21.06.20 17:24:11.268
  CMD_T_RUN msg_body:
[20000a70,20000af0] sz=128
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Start Process
Drive_T_Start cmd=0x80
StartTCommand cmd=0x80
  1 Status byte = 0x02
Drive_T_Start retcode8=0x00 stat_byte=0x02 diag_byte=0x00
Finish Process
Drive_T_Termination rc=0x00000000
< msg_send_res rc=0x00000200, data len=0
< msg_send_res opcode=0x05 body_len=1032
[20000650,200006d0] sz=128
00080305 00000200 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

Рисунок 32 – вывод отладочной информации по команде «Выдать состояние».

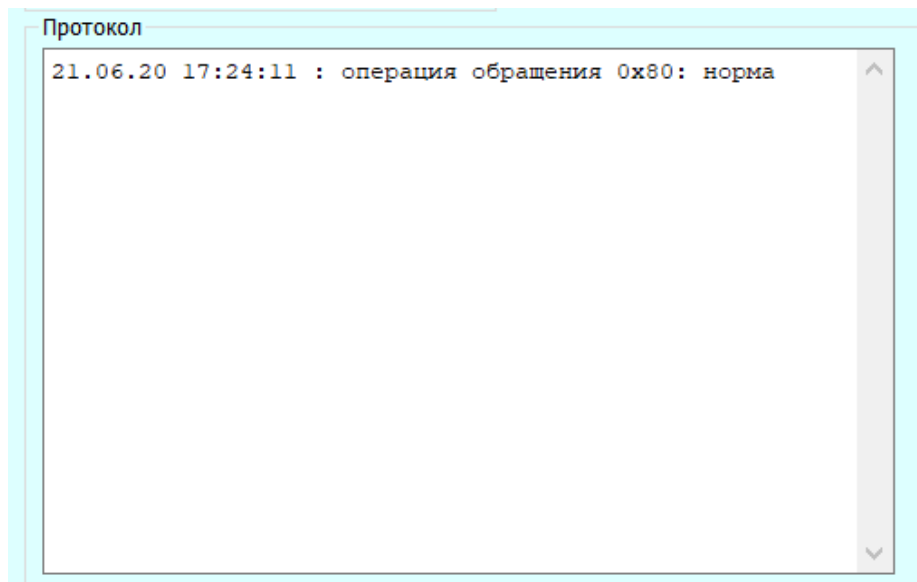


Рисунок 33 – результат выполнения команды «Выдать состояние».

4.2.5 Команда «Вывод»

Команда описана в [12]. В проекте реализована эмуляция этой команды. Команда «Вывод» записывает в ОЗУ МК массив байт. Результаты работы команды приведены на рисунках 34, 35.

```
21.06.20 17:26:35.924
> tcp_get len=76 opcode=0x6 , seq=4 bodylen=72
< ack_send code=0x0
< dev_form_msg 0xAA msg_body_len=4
21.06.20 17:26:36.417
CMD_T_RUN msg_body:
[20000a70,20000af0] sz=128
00000008 00000040 A4A3A2A1 A8A7A6A5 ACABAAA9 B0AFAEAD B4B3B2B1 B8B7B6B5
BCBBBAB9 C0BFBEED C4C3C2C1 C8C7C6C5 CCCBCAC9 D0CFCECD D4D3D2D1 D8D7D6D5
DCDBDAD9 E0DFDEDD 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Start Process
Drive_T_Start cmd=0x08
StartTCommand cmd=0x8
et_Write len = 64
et_Write data = 0xA1
1 Status byte = 0x02
et_Write data: (len=64)
[20103000,20103040] sz=64
A4A3A2A1 A8A7A6A5 ACABAAA9 B0AFAEAD B4B3B2B1 B8B7B6B5 BCBBBAB9 C0BFBEED
C4C3C2C1 C8C7C6C5 CCCBCAC9 D0CFCECD D4D3D2D1 D8D7D6D5 DCDBDAD9 E0DFDEDD

2 Status byte = 0x01
Drive_T_Start retcode8=0x00 stat_byte=0x01 diag_byte=0x00
Finish Process
Drive_T_Termination rc=0x00000000
< msg_send_res rc=0x00000100, data len=0
< msg_send_res opcode=0x05 body_len=1032
[20000650,200006d0] sz=128
00080405 00000100 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Рисунок 34 – вывод отладочной информации по команде «Вывод».

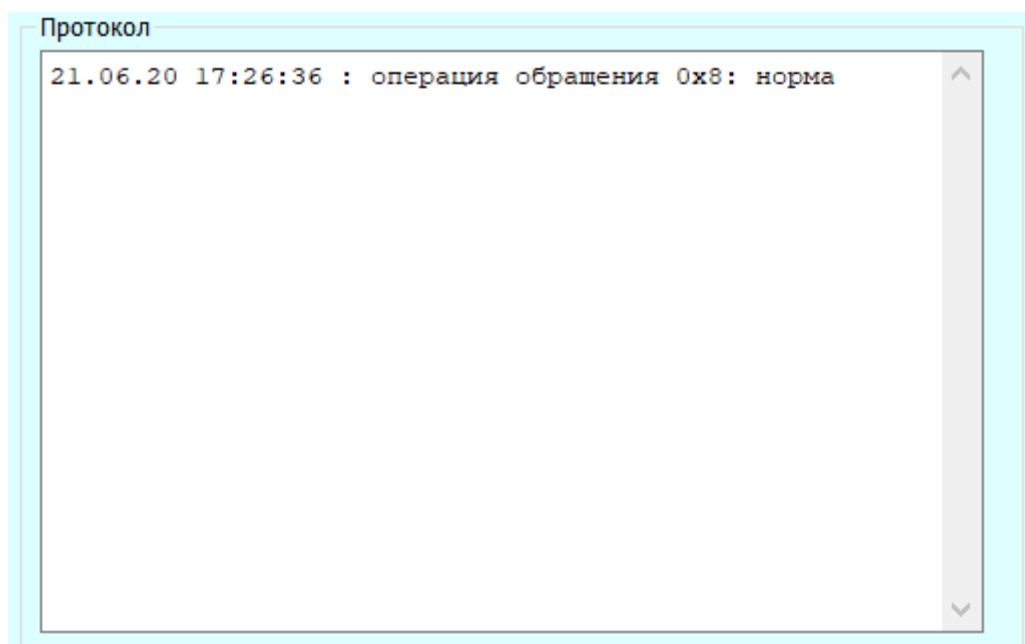


Рисунок 35 – результат выполнения команды «Вывод».

4.2.6 Команда «Ввод»

Команда описана в [12]. В проекте реализована эмуляция этой команды. Команда «Ввод» считывает из ОЗУ МК массив байт. Результаты работы команды приведены на рисунках 36, 37.

```

21.06.20 17:28:46.132
> tcp_get len=12 opcode=0x6 , seq=5 bodylen=8
< ack_send code=0x0
< dev_form msg 0xAA msg_body_len=4
21.06.20 17:28:46.547
CMD_T_RUN msg_body:
[20000a70,20000af0] sz=128
00000010 00000040 A4A3A2A1 A8A7A6A5 ACABAAA9 B0AFAEAD B4B3B2B1 B8B7B6B5
BCBBBAB9 C0BFBEED C4C3C2C1 C8C7C6C5 CCCBCAC9 D0CFCECD D4D3D2D1 D8D7D6D5
DCDBDAD9 E0DFDEDD 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Start Process
Drive_T_Start cmd=0x10
StartTCommand cmd=0x10
1 Status byte = 0x03
2 Status byte = 0x02
et_Read data: (len=64)
[20100800,20100840] sz=64
A4A3A2A1 A8A7A6A5 ACABAAA9 B0AFAEAD B4B3B2B1 B8B7B6B5 BCBBBAB9 C0BFBEED
C4C3C2C1 C8C7C6C5 CCCBCAC9 D0CFCECD D4D3D2D1 D8D7D6D5 DCDBDAD9 E0DFDEDD

3 Status byte = 0x01
Drive_T_Start retcode8=0x00 stat_byte=0x01 diag_byte=0x00
Finish Process
Drive_T_Termination rc=0x00000000
< msg_send_res rc=0x00000100, data len=64
< msg_send_res opcode=0x05 body_len=1032
[20000650,200006d0] sz=128
00480505 00000100 00000040 A4A3A2A1 A8A7A6A5 ACABAAA9 B0AFAEAD B4B3B2B1
B8B7B6B5 BCBBBAB9 C0BFBEED C4C3C2C1 C8C7C6C5 CCCBCAC9 D0CFCECD D4D3D2D1
D8D7D6D5 DCDBDAD9 E0DFDEDD 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

Рисунок 36 – вывод отладочной информации по команде «Ввод».

```

Протокол
21.06.20 17:28:46 : операция обращения 0x10: норма
21.06.20 17:28:46 : считать данные
21.06.20 17:28:46 : -----
00000000      A4A3A2A1  A8A7A6A5  ACABAAA9  B0AFAEAD
00000010      B4B3B2B1  B8B7B6B5  BCBABAB9  C0BFBEED
00000020      C4C3C2C1  C8C7C6C5  CCCBCAC9  D0CFCECD
00000030      D4D3D2D1  D8D7D6D5  DCDBDAD9  E0DFDEDD

```

Рисунок 37 – результат выполнения команды «Вывод».

4.2.7 Команда «Диагностика»

Команда описана в [12]. В проекте реализована эмуляция этой команды. Команда «Диагностика» выдает последний сформированный диагностический байт [12]. Результаты работы команды приведены на рисунках 38, 39, 40.

```

21.06.20 17:31:05.745
> tcp_get len=12 opcode=0x6 , seq=6 bodylen=8
< ack_send code=0x0
< dev_form_msg 0xAA msg_body_len=4
21.06.20 17:31:06.686
CMD_T_RUN msg_body:
[20000a70,20000af0] sz=128
000000C0 00000000 A4A3A2A1 A8A7A6A5 ACABAAA9 B0AFAEAD B4B3B2B1 B8B7B6B5
BCBABBAB9 C0BFBEED C4C3C2C1 C8C7C6C5 CCCBCAC9 D0CFCECD D4D3D2D1 D8D7D6D5
DCDBDAD9 E0DFDEDD 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Start Process
Drive_T_Start cmd=0xC0
StartTCommand cmd=0xC0
et_Diagnostics diag_byte = 0x91
Drive_T_Start retcode8=0x00 stat_byte=0x01 diag_byte=0x91
Finish Process
Drive_T_Termination rc=0x00000000
< msg_send_res rc=0x00910100, data len=0
< msg_send_res opcode=0x05 body_len=1032
[20000650,200006d0] sz=128
00080605 00910100 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

Рисунок 38 – вывод отладочной информации по команде «Диагностика».

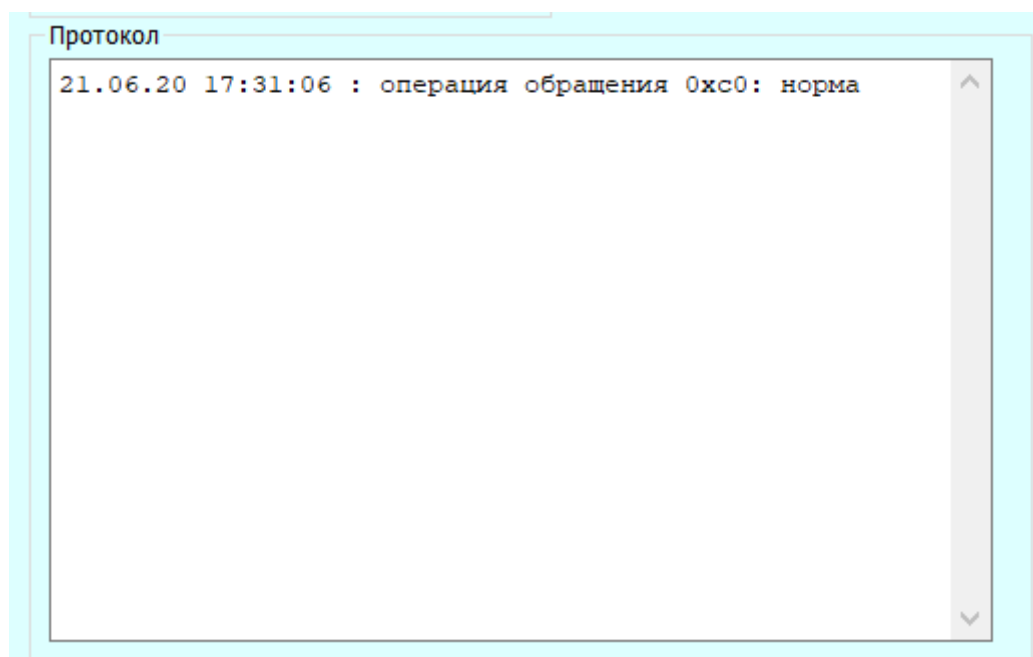


Рисунок 39 – результат выполнения команды «Диагностика».

гс	0x0
б.сост.	0x1
б.диаг.	0x91

Рисунок 40 – результат выполнения команды «Диагностика».

5 Безопасность жизнедеятельности

В современном мире вычислительная техника, в том числе и ПЭВМ, стали неотъемлемой частью в различных сферах деятельности. Некоторые сферы деятельности напрямую связаны с применением ПЭВМ и человеку, которым занимается этой деятельностью приходится проводить за ПЭВМ большую часть времени. Невозможно полностью избежать вредоносных факторов при работе с вычислительной техникой, в том числе с ПЭВМ, однако если соблюдать определенные рекомендации пользования ПЭВМ, вредные факторы можно существенно минимизировать. Основными правилами являются соблюдение правил безопасности, правильное обустройство рабочего места и правильное расположение техники.

Для уменьшения психофизического фактора (стресс, монотонность работы, долгое пребывание в одном положении) желательно выполнять следующие рекомендации:

- Делать перерыв каждый час – два на 15-20 минут;
 - Делать небольшую физическую разгрузку (небольшая прогулка, простые упражнения);
 - Рекомендуется использовать ЖК дисплеи с высоким разрешением
- В ГОСТ 12.2.032 78 прописаны основные требования к организации рабочего места и его оборудованию:
- Высота рабочей поверхности стола должна регулироваться в пределах 680-800 мм;
 - Параметры рабочего стола: ширина – 800, 1200 или 1400 мм, глубина – 800 или 1000
 - Пространство для ног: высота – не менее 600 мм; ширина – не менее 500 мм
 - рабочий стул (кресло) должен быть подъемно-поворотным и регулируемым по высоте, углам наклона, а также расстоянию спинки до сидения;

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		50

ПЭВМ работает преимущественно с напряжением 220 В, которое предоставляет опасность для человека. Требования безопасности, при работе с ПЭВМ прописаны в ГОСТ 12.2.007-75.

					ПГУ 2.090301.37.001 ПЗ	Лист
						51
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В ходе выполнения данной выпускной квалификационной работы были изучены способы сетевых взаимодействий на уровне транспортного уровня стека протоколов TCP/IP для встраиваемых систем, были получены навыки программирования на языке C, Python. Были получены навыки построения программ для МК с использованием ОСРВ.

Результатом решения поставленной задачи можно считать разработанный и протестированный комплекс программ для взаимодействия с накопителем (эмуляцией) по ГОСТ Р 50451-92. Разработанное программное обеспечение предоставляет разработчику шаблон для будущей разработки полнофункциональной специализированного устройства для взаимодействия с реальным накопителем.

В дальнейшем программа может быть улучшена путём уменьшения реакции системы на событие.

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		52

Список использованных источников

1. Д. Ритчи, Б. Керниган. Язык программирования С, второе издание, 2016 г.
2. FreeRTOS, справочное руководство – URL:
https://www.freertos.org/Documentation/RTOS_book.html
3. Mastering the FreeRTOS Real Time Kernel – a Hands On Tutorial Guide – URL: https://www.freertos.org/Documentation/RTOS_book.html
4. Олифер В., Олифер Н. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 5-е изд. – СПб.: Питер, 2016 г.
5. Спецификация на микросхему 1986BE1T, 1986BE1AT – URL:
https://ic.milandr.ru/products/mikrokontrollery_i_protssory/32_razryadnye_mikrokontrollery/1986vekh_avia/1986ve1t
6. Форум АО «ПКК Миландр» - URL: <https://forum.milandr.ru/>
7. Руководство по применению uIP – URL:
<http://microsin.net/Download.cnt/doc/uIP/index.html>
8. Д. Браун. Discovering the STM32 Microcontroller, 2013 г.
9. М. Лутц. Изучаем Python (том 1,2), пятое издание, 2019 г.
10. М. Лутц. Программирование на Python (том 1, 2), четвертое издание, 2011 г.
11. В. Дронов, Н. Прохоренок. Python 3 и PyQt 5. Разработка приложений, 2016 г.
12. ГОСТ Р 50451-92. Система интерфейсов для сопряжения радиоэлектрических средств. Интерфейс Т. Общие требования, Госстандарт России, Москва.

Приложение А. Фрагменты листингов программы

Приложение А1. Листинг программы управления.

Файл «PVM.py»

```
from PyQt5 import QtGui, uic, QtCore
from PyQt5.QtWidgets import QMainWindow, QApplication
from PyQt5.QtGui import QPalette, QColor

from my_thread import MyThread
from eeprom_net_set import *
from net_cfg_ini import *
from my_utils import *

from pvm_cmd import *
import pvm

IP_DEFAULT = '192.168.101.80'
PORT_TCP1 = 7139
te_MAX_LINES = 50

class MyMain(QMainWindow):

    def __init__(self, *args):
        super(MyMain, self).__init__(*args)
        uic.loadUi('pvm.ui', self)
        # uic.loadUi('mtel.ui', self)

        self.setWindowTitle(self.windowTitle() + '. ' + _ts_ +
                               _cr_)

        # self.init_tips()
        ack_tip = pvm.ack_tip
        self.le_ack.setToolTip(ack_tip)
        self.label_MesRes.setToolTip(ack_tip)

        # self.init_widgets()

        # подключение кнопок к сигналам
        self.init_widget_connect()

        p = self.radioButton.palette()
        p.setColor(QPalette.Button, QColor("White")) #
        ("lightGreen") ("Green")
        self.radioButton.setPalette(p)
        self.radioButton.setDown(True)
        # self.radioButton.setChecked(True)
        self.radioButton.setPalette(p)
        self.radioButtonInvisible.setVisible(False)
```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		54

```

# инициализация переменных для хранения IP и PORT
self.init_ip_and_port()

# IP из конфигурации
self.read_ini()
print ('pvm.ini')
print (self.ip)

# исходное состояние программы
self.cnt = 0 # msg count
self.cnt1 = 0 # msg count
self.seq = 0 # cmd count

self.cnt_max = te_MAX_LINES

self.mon = False;
self.tcp_flag = False
self.tcp_sent = False
# self.tcp_wait_reply = False не помогает

self.change_port()
self.change_color()

self.set_pos_checkBoxes(not self.pos, not self.repeat)

self.groupBoxCmd.setEnabled(False)
self.groupBoxParmWrite.setEnabled(False)

self.socket_err = False

...
# end of __init__

# инициализация comboBox
def init_comboBoxes(self):
    self.parm_nak_codes= pvm.parm_nak_codes
    self.parm_nak_codes=dev.parm_nak_codes

    self.comboBox_parm_nak.addItem(pvm.comboBox_parm_nak_txt)

    self.l_mode_nak.setText(hex(pvm.parm_nak_codes[0]))
# инициализация переменных для IP и PORT
def init_ip_and_port(self):

    self.peerIP = IP_DEFAULT
    self.peerPort = PORT_UDP

    self.ip = IP_DEFAULT

    self.port_udp = PORT_UDP
    self.port_tcp1 = PORT_TCP1
    self.port_tcp2 = PORT_TCP2

```

```

# вызов диалога ini
QtCore.pyqtSlot()
def on_pb_ini_cfg_clicked(self):
    del self.my_ini
    os.system('pvm_net_cfg.py')
    #os.wait() # ?
    self.read_ini()
    self.change_ip()
    self.change_port()

# чтение ini
def read_ini(self):
    self.my_ini = MyIni('pvm.ini','pvm_default.ini')
    ips = self.my_ini.get_ip_s()

    self.ip = ips[0]

    ports = self.my_ini.get_port_s()
    print (ports)

    self.port_udp = int(ports[0])

    self.port_tcp1 = int(ports[3])

def set_buttons_tcp_enable(self):
    self.pb_ssn_open.setEnabled(True)
    self.pb_ssn_close.setEnabled(False)

def set_buttons_tcp_disable(self):
    self.pb_ssn_open.setEnabled(False)
    self.pb_ssn_close.setEnabled(False)

# послать команду и получить на нее квитанцию
def send_cmd_and_receive_ack(self,code,seq, length, body):

    if self.tcp_flag:
        msg = dev.pack_msg(code, seq, length, body)
        # print(msg)
        print(bytes2str_n(msg, 8))

        self.s.sendall(msg)
        print('> The client send ', msg)

        self.tcp_sent = True
        # self.tcp_wait_reply = True

        reply = self.s.recv(dev.HEADER_LEN + dev.MSG_ACK_LEN)
        print('< The server said ', repr(reply))

        code_b, seq, length = dev.extract_header(reply)
        txt = 'header: code=' + str(code_b) + ',seq=' + str(seq) +
        ',body len=' + str(length)

```

```

print(txt)

print('receive (waited) ACK:')

body = dev.extract_body(reply)
print('body:', body)

ack_code = int.from_bytes(body, byteorder='little')
print('ack=', '0x{:08X}'.format(ack_code))
# ack_code = body[0:1]
# print(ack_code)

else:
    print('no tcp')
    self.l_read_status.setText('no connection')
    ack_code = -2

self.show_ack(ack_code)
# self.tcp_wait_reply = False

return ack_code

# получить ответное сообщение на команду
def receive_reply_msg(self):
    # self.tcp_wait_reply = True
#     reply = self.s.recv(1024+8)

    if self.socket_err:
        self.s.recv(2048)

    try:
        reply = self.s.recv(2048)
        print('< The server said len=', len(reply),
repr(reply))

    except socket.error as e:
        print('Error receiving data ',e)
        body = ''
        self.socket_err = True
        return (body)

    self.socket_err = False

    code_b, seq, length = dev.extract_header(reply)
    txt = 'header: code=' + str(code_b) + ',seq=' + str(seq) +
',body len=' + str(length)
    print(txt)

    body = dev.extract_body(reply)
    print('body:', body)

    return (body)

```

```

# отображение цвета результата выполнения команды
def show_run_rc(self, boolean_res):
    p = self.l_run_rc.palette()
    if (boolean_res):
        color = "darkGreen"
    else:
        color = "darkRed"
    p.setColor(self.l_run_rc.foregroundRole(), QColor(color))
    self.l_run_rc.setPalette(p)

# отображение цвета радио кнопки квитанции
def show_rb_ack(self, active, on):
    p = self.radioButtonMesRes.palette()
    if (active):
        if (on):
            color = "Green"
        else:
            color = "Red"
    else:
        color = "White"
    p.setColor(QPalette.Button, QColor(color))
    self.radioButtonMesRes.setPalette(p)
    self.radioButtonMesRes.setDown(True)

# отображение квитанции
def show_ack(self, rc):
    print('ack=', '0x{:02X}'.format(rc))
    ack_msg = 'квитанция: 0x{:02X}'.format(rc)
    self.le_ack.setText(ack_msg)
    if not(rc==0):
        ack_msg = 'ошибка ' + ack_msg
    self.statusBar().showMessage(ack_msg)

    if (rc==0):
        self.show_rb_ack(1, True)
    else:
        self.show_rb_ack(1, False)

"""
-----
Device
-----
"""

def show_state(self, body):
    print('show state')
    # print('body:', body)
    status_b = body[4:8]
    status_max_block = body[8:8 + 4]
    status_retcode = body[12:12 + 4]
    status_sense_word = body[16:16 + 4]
    blk_word = body[20:20 + 2]

```

```

        k_word    = body[22:22 + 2]

        max_block = int.from_bytes(status_max_block,
byteorder='little')
        retcode = int.from_bytes(status_retcode, byteorder='little')
        sense_word = int.from_bytes(status_sense_word,
byteorder='little')
        blk = int.from_bytes(blk_word, byteorder='little')
        k    = int.from_bytes(k_word, byteorder='little')

        print('status_b=', str(status_b))

        print('status=', '0x{:08X}'.format(status), 'ready=',
status_ready,
            'T=', status_T, 'busy=', status_busy)

        print('max_block=', max_block)
        print('retcode=', '0x{:08X}'.format(retcode))
        print('blk range : blk=', blk)
        print('blk range : k=', k)

        res_str = body[0:4].decode()
        self.l_read_status.setText(res_str + '
0x{:08X}'.format(status))
        self.l_ready.setText(str(status_ready))
        self.l_T.setText(str(status_T))
        self.l_busy.setText(str(status_busy))
        self.le_block_max.setText(str(max_block))

        self.sb_block_start_t.setValue(blk)
        self.sb_block_count_t.setValue(k)

        self.repeat = status_repeat

        self.set_pos_checkBoxes(not self.pos, not self.repeat)

# прочесть состояние прибора
@QtCore.pyqtSlot()
def on_pb_read_state_clicked(self):
    msg = "pb_read_state " + str(self.seq)
    print(msg)
    self.statusBar().showMessage(msg)

    ack = self.send_cmd_and_receive_ack(DEV_CMD_READ_STATUS,
self.seq, 0, 0)
    print('ack=', ack)
    if (ack == 0):
        body = self.receive_reply_msg()
        if (body == ''):
            m = 'timeout receiving reply message'
            print(m)
            self.l_run_rc.setText(m)

```

```

        self.show_run_rc(False)
        ret = False
        return ret

    # print('body:', body)
    ret = True

    if ret == True:
        self.show_state(body)
    else:
        self.l_read_status.setText('?')

    self.seq += 1
    return ret

def generate_test_data(self, datalen):
    import zlib

    data = bytearray(datalen)
    DATA_FIRST = 0xA1
    for i in range(0, len(data)):
        data[i] = (DATA_FIRST + i) & 0xFF

    print('данные для записи:')
    print('bytes: ', len(data))
    print_array_byteX(data, 0, 0)
    # data_crc32 = zlib.crc32(data)
    # print('crc32 ', int_to_HEX(data_crc32))
    # print()
    return (data)

# разбор ответного входного сообщения
def parse_msg_res_t(self, body):
    code = int(self.l_mode_vnd_t.text(), 16)
    datalen_b = (body)[4:8]
    datalen = int.from_bytes(datalen_b, byteorder='little')
    print('data len=', datalen)
    # ввод
    if (code == 0x10):
        txt= 'считать данные'
        print(txt)
        data = body[8:8 + datalen]
        str1 = dump_str(data, 0, 4)
        print(str1)
        self.msg_log(txt)
        self.msg_log(str1)

# упаковка команды для T
def pack_body_t(self):
    print("pack body vndt")
    code = int(self.l_mode_vnd_t.text(), 16)
    #ВЫВОД, ВВОД
    if (code==0x08 or code==0x10):

```

```

        datalen = int(self.sb_datalenT.value())
        # block = int(self.sb_block_start_t.value())
        # block_count = int(self.sb_block_count_t.value())
    else:
        datalen = 0;
        # block = 0;
        # block_count = 0;
    if (code==0x08):
        # if block_count==3:  # а нужно ли здесь это
        #     block_count = 2 # а нужно ли здесь это
        data = self.generate_test_data(datalen)
    else:
        data = ''
    print('code=', hex(code), 'data=', data, 'datalen=', datalen)
    res = b"\x00\x00\x00"
    # res = 0
    # if (code == 0x08):
    #     msg_body = struct.pack('<1H', code) + \
    #                 struct.pack('<1H', res) + \
    #                 struct.pack('<1I', datalen) + \
    #                 data
    # else:
    #     msg_body = struct.pack('<1H', code) + \
    #                 struct.pack('<1H', res) + \
    #                 struct.pack('<1I', datalen) + \
    #                 data.encode()

    if (code == 0x08):
        msg_body = struct.pack('<1B', code) + \
                    res + \
                    struct.pack('<1I', datalen) + \
                    data
    else:
        msg_body = struct.pack('<1B', code) + \
                    res + \
                    struct.pack('<1I', datalen) + \
                    data.encode()

    print('data_len=', len(data))
    # msg_len = len(msg_body)
    print(msg_body, 'msg body len=', len(msg_body))
    return msg_body

# установить новый порт, IP-адрес, MAC
@QtCore.pyqtSlot()
def on_pb_IP_clicked(self):
    msg = "pb IP " + str(self.seq)
    print(msg)
    self.statusBar().showMessage(msg)

    port = int(self.le_port.text())

    ip_addr_list = list(self.le_ip_addr_2.text().split('.'))

```

```

ip_addr = [int(val) for val in ip_addr_list]
print(ip_addr_list)
buf_ip = bytes(ip_addr)

from re import sub
mac_addr = sub(r"[:]", " ", self.le_mac.text())
print(mac_addr)
buf_mac = bytearray.fromhex(mac_addr)

msg_body = struct.pack('<1H', port) +\
                buf_ip +\
                buf_mac

ack = self.send_cmd_and_receive_ack(DEV_CMD_SET_IP_ADDR,
self.seq, len(msg_body), msg_body)
print('ack=', ack)
if (ack == 0):
    ...
else:
    self.l_read_status.setText('?')
    self.seq += 1

# ВЫПОЛНИТЬ КОМАНДУ T
@QtCore.pyqtSlot()
def on_pb_t_run_clicked(self):
    msg = "pb_t_run " + str(self.seq)
    print(msg)
    self.statusBar().showMessage(msg)

    if not self.check_parm_vndt():
        print("ошибка параметров команды ")
        return

    code = int(self.l_mode_t.text(), 16)
    body = self.pack_body_t()
    ack = self.send_cmd_and_receive_ack(DEV_CMD_VNDT_RUN,
self.seq, len(body), body)
    if (ack == 0):
        body = self.receive_reply_msg()
        retcode_b = (body)[0:1]
        stat_b     = (body)[1:2]
        diag_b     = (body)[2:3]
        datalen_b = (body)[4:8]

        retcode = int.from_bytes(retcode_b, byteorder='little')
        datalen = int.from_bytes(datalen_b, byteorder='little')
        stat = int.from_bytes(stat_b, byteorder='little')
        diag = int.from_bytes(diag_b, byteorder='little')
        print('retcode=', retcode_b)
        print('stat=', stat_b)
        print('diag=', diag_b)

        datalen = int.from_bytes(datalen_b, byteorder='little')

```

```

        print('datalen=',datalen) #datalen= 1032 ?
        self.l_run_rc_t.setText(hex(retcode))
        self.l_run_sb_t.setText(hex(stat))
        self.l_run_db_t.setText(hex(diag))
        self.l_run_rc.setText(hex(retcode))

        if (retcode == 0):
            txt='операция обращения к накопителю' + hex(code) + ':'
норма'
            print(txt)
            self.msg_log(txt)
            self.parse_msg_res_t(body)
            bool_rc = True
        else:
            bool_rc = False
            self.show_run_rc(bool_rc)

    else:
        m = 'ack err'
        print(m)
        self.l_run_rc.setText(m)
        self.show_run_rc(False)

    self.seq += 1

# установить диапазон блоков (T)
@QtCore.pyqtSlot()
def on_pb_set_blk_range_t_clicked(self):
    msg = "pb set blk range t " + str(self.seq)
    print(msg)
    self.statusBar().showMessage(msg)

    blk = self.sb_block_start_t.value()
    k = self.sb_block_count_t.value()
    print ('blk=',blk,'k=',k)

    msg_body = struct.pack('<1H', blk) + struct.pack('<1H', k)

    ack = self.send_cmd_and_receive_ack(DEV_CMD_T_SET_BLK,
self.seq, len(msg_body), msg_body)
    print('ack=', ack)
    if (ack == 0):
        ...
    else:
        self.l_read_status.setText('?')
        self.seq += 1

# проверка соединения : отправка и получение тестового пакета
def test_connection(self):
    test_data = 'abcdefghijklmnopqrstuvwxy1234567890'

    print('> The client send ', test_data.encode())

```

```

        ack = self.send_cmd_and_receive_ack(DEV_CMD_TEST_CONNECTION,
self.seq, len(test_data), test_data.encode())
        print ('ack=',ack)
        if (ack == 0):
            body = self.receive_reply_msg()
            print('body:', body)
            if test_data == body.decode():
                res = True
            else:
                res = False
        else:
            res = False

        self.seq += 1

        return (res)

# команда проверка соединения :  отправка и получение тестового
пакета
@QtCore.pyqtSlot()
def on_pb_check_connection_clicked(self):
    print ('test connection')
    msg = "test connection " + str(self.cnt)
    self.l_status.setText(msg)
    self.statusBar().showMessage(msg)

    self.cnt += 1

    if self.tcp_flag:
        print('tcp')
        ret = self.test_connection()
        if ret:
            res_str = 'норма'
            rc = 0
        else:
            res_str = 'ошибка'
            rc = 0x21

        print (res_str)
        self.l_check_connection.setText(res_str)
        self.l_check_connection_2.setText(str(self.cnt))
        self.statusBar().showMessage(res_str)

    else:
        print('no tcp')
        self.l_check_connection.setText('???')
        self.l_check_connection_2.setText(' ')

def show_fw(self,body):
    self.pte_status.clear()
    self.msg_log('< ' + str(body));
    stat_txt = info(body)

```

```

        print(stat_txt)
        self.pte_status.appendPlainText(stat_txt)

# чтение инфо прошивки (id, timestamp, crc)
@QtCore.pyqtSlot()
def on_pb_read_fw_clicked(self):
    print('on_pb_read_fw_clicked')
    msg = "pb_read_fw" + str(self.cnt)
    self.l_status.setText(msg)
    self.statusBar().showMessage(msg)

    ack = self.send_cmd_and_receive_ack(DEV_CMD_READ_FW, self.seq,
0, 0)
    if (ack == 0):
        body = self.receive_reply_msg()
        if (body=='') :
            m = 'timeout receiving reply message'
            print(m)
            self.l_run_rc.setText(m)
            self.show_run_rc(False)
            return;
        else:
            self.show_fw(body)
            self.seq += 1

def show_time(self,body):
    self.pte_status.clear()
    self.msg_log('< ' + str(body));
    x = int.from_bytes(body, byteorder='little')
    print(str(x))
    stat_txt = 'time ' + str(x) + ' c'
    print(stat_txt)
    self.pte_status.appendPlainText(stat_txt)

# чтение времени прибора
@QtCore.pyqtSlot()
def on_pb_read_time_clicked(self):
    print('on_pb_read_time_clicked')
    msg = "pb_read_time " + str(self.cnt)
    self.l_status.setText(msg)
    self.statusBar().showMessage(msg)
    self.seq += 1

    ack = self.send_cmd_and_receive_ack(DEV_CMD_READ_TIME,
self.seq, 0, 0)
    print('ack=', ack)
    if (ack == 0):
        body = self.receive_reply_msg()
        if (body==""):
            m = 'timeout receiving reply message'
            print(m)
            self.l_run_rc.setText(m)
            self.show_run_rc(False)
            return False

```

```

        else:
            # print('body:', body)
            self.show_time(body)
            return True
    else:
        return False

# перезагрузка прибора
@QtCore.pyqtSlot()
def on_pb_reset_clicked(self):
    msg = "pb_reset_2 " + str(self.seq)
    print(msg)
    self.statusBar().showMessage(msg)

    ack = self.send_cmd_and_receive_ack(DEV_CMD_RESET, self.seq,
0, 0)
    print('ack=', ack)
    if (ack == 0):
        time.sleep(1.0)

        self.on_pb_ssn_close_clicked()

"""
-----
настройки сети программы
-----
"""
# новый IP
@QtCore.pyqtSlot()
def on_pb_new_ip_assign_clicked(self):
    self.peerIP = self.le_ip_addr.text()
    print('new peer IP', self.peerIP)
    self.peerPort = int(self.le_port_ctrl.text())
    print('new peer Port', self.peerPort)
    #new_ip(newip)
    self.rb_new_ip_assigned.setChecked(True)
    self.change_color()

# новый IP Localhost
@QtCore.pyqtSlot()
def on_pb_localhost_assign_clicked(self):
    self.le_ip_addr.setText('127.0.0.1')
    self.pb_new_ip_assign_click()

def le_ip_port_changed(self):
    self.new_ip_assigned.setChecked(False)
    print('changed')

"""
# -----
# TCP
# -----
"""

```

```

def on_open_action(self,reply):
    body = self.extract_reply_msg_body(reply);
    self.show_state(body)

@QtCore.pyqtSlot()
def on_pb_ssn_open_clicked(self):
    msg = "ssn open " + str(self.cnt)
    self.l_status.setText(msg)
    self.statusBar().showMessage(msg)

    self.show_rb_ack(False, False)
    self.seq = 0

    self.tcp_flag = True

    timeout = self.sb_timeout.value()/1000.
    print ('timeout=',timeout)

#    print (type(self.peerIP), self.peerIP)
    try:
        self.s = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
    except socket.error as e:
        print('Error creating socket ', e)
        return
    print('Connect to', self.peerIP, self.peerPort)
    self.s.settimeout(timeout)
    try:
        self.s.connect(( self.peerIP, self.peerPort))
    except socket.gaierror as e:
        print('Address-related error connecting to server
',e)

        return
    except socket.error as e:
        print('Connecting error',e)
        self.statusBar().showMessage('Connecting error')
        return
    print('Client has been assigned socket name',
self.s.getsockname())
    try:
        reply = self.s.recv(256)
        print('< The server said ', repr(reply))
        self.pb_ssn_close.setEnabled(True)
        self.pb_ssn_open.setEnabled(False)
        self.groupBoxCmd.setEnabled(True)
        self.on_open_action(reply)
    except socket.error as e:
        print('Error receiving data ',e)

@QtCore.pyqtSlot()
def on_pb_ssn_close_clicked(self):
    msg = "ssn closed " + str(self.cnt)
    self.l_status.setText(msg)

```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		67

```

self.statusBar().showMessage(msg)

self.show_rb_ack(False, False)

self.pb_ssn_close.setEnabled(False)
self.pb_ssn_open.setEnabled(True)

self.groupBoxCmd.setEnabled(False)
# self.pb_get_info.setEnabled(False)
# self.pb_status.setEnabled(False)
# self.pb_get_time.setEnabled(False)

self.groupBoxT.setEnabled(False)

if self.tcp_flag == False :
    ...
    return

print('> закрыть соединение')
self.s.shutdown(socket.SHUT_RDWR);
self.tcp_sent = False

print('закрываем локальное соединение ', self.s.getsockname())
self.s.close()
self.tcp_flag = False
print('соединение закрыто')

def closeEvent(self, event):
    self.hide()
    self.mythread.running = False
    self.mythread.wait(5000)
    event.accept()
    # Вызывается при закрытии окна
    # Скрываем окно
    # Изменяем флаг выполнения
    # Даем время чтобы закончить
    # Закрываем окно

@QtCore.pyqtSlot()
def on_pb_close_clicked(self):
    self.hide() # Скрываем окно
    # self.mythread.running = False # Изменяем флаг выполнения
    # self.mythread.wait(5000) # Даем время чтобы закончить
    self.close()

# конец методов MyMain

app = QApplication(sys.argv)
widget = MyMain()
widget.show()
#print('iu.py: завершение', time.strftime("%d.%m.%y %H:%M:%S"))
sys.exit(app.exec_())

```

Файл «dev_cmd.py»

```
DEV_CMD_READ_STATUS          = 0x01
DEV_CMD_TEST_CONNECTION      = 0x02
DEV_CMD_VNDT_SET_BLK         = 0x03
DEV_CMD_VNDT_RUN              = 0x04
DEV_CMD_READ_FW               = 0x11
DEV_CMD_READ_TIME             = 0x12
DEV_CMD_SET_IP_ADDR           = 0x20
DEV_CMD_RESET                 = 0x40
DEV_CMD_ERR                   = 0xEE
```

Файл «net_cfg_ini.py»

```
import time

from my_utils import *

class MyIni():
    def __init__(self, filename, default_filename):
        self.filename = filename
        self.default_filename = default_filename
        #print('ini',self.filename)

        self.origconfig = ConfigParser()
        self.config = ConfigParser()
        self.defaultconfig = ConfigParser()

        # parse existing file
        self.origconfig.read(self.filename)
        self.config.read(self.filename)
        #self.defaultconfig.read_dict(DEFAULT_CONFIG )
        self.defaultconfig.read(self.default_filename)

    def get_ip(self, unit, ipdef):
        ip = self.config[unit][ipdef]
        ip = ip_validate(ip, DEFAULT_CONFIG[unit][ipdef]);
        return(ip)

    def set_ip(self, unit, ipdef, ip ):
        self.config[unit][ipdef] = ip

    def get_port(self,portdef):
        port = self.config['ports'][portdef]
        return(port)

    def set_port(self, portdef, port ):
        self.config['ports'][portdef] = port

    def get_ip_s(self): # возвращает список из 4-х iP

        ip_list = []
```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.37.001 ПЗ

Лист

69

```

        for unit in ['unit1','unit2']:
            for name, value in self.config.items(unit):
                value, corr = ip_validate(value,
self.defaultconfig[unit][name]);
                ip_list.append(value)
        return(ip_list)

# проверить на корректность адреса IP в списке IP
def ip_validate(self,ips):
    print ("enter function: %s" % inspect.stack()[0][3])
    ip_list = []
    ip_list_default = self.get_ip_s_default()
    correct_list = []
    i = 0
    for ip in ips:
        ip, is_correct = ip_validate(ip, ip_list_default[i]);
        ip_list.append(ip)
        #print(ip, ip_list)
        correct_list.append(is_correct)
        i = i + 1
    return(ip_list,correct_list)

def get_time_update(self):
    return(self.origconfig['general']['time_upd'])

# обновить файл конфигурации
def update(self):
    print ("enter function: %s" % inspect.stack()[0][3])
    upd = self.config != self.origconfig
    print('is update? ', upd)

    if upd:

        print('new config')
        t = time.strftime("%d.%m.%y %H:%M:%S")
        self.config['general']['time_upd'] = t
        #self.print();

        # save to a file
        with open(self.filename, 'w') as configfile:
            self.config.write(configfile)

        configfile.close()
    else:
        #print()
        #print('config actual')
        ...
    return(upd)

# с классом MyIni
def test_ip_validate():
    print(time.strftime("%d.%m.%y %H:%M:%S"))

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.37.001 ПЗ

Лист

70

```

my_ini = MyIni('iu.ini','iu_default.ini')

test_ip = '256.255.1.1'
test_ip, is_correct = ip_validate(test_ip, '192.168.101.80');
print(test_ip, is_correct)

def test_ini2():
    print(time.strftime("%d.%m.%y %H:%M:%S"))

    my_ini = MyIni('iu.ini','iu_default.ini')

    ip11,ip12,ip21,ip22 = my_ini.get_ip_s()
    print (ip11,ip12,ip21,ip22)
    ##      udp1,udp2 = my_ini.get_port_s()
    ##      print (udp1,udp2)

    my_ini.update_general()
    #my_ini.update_ip_s(ip11,ip12,ip21,ip22)
    my_ini.update()

    #my_ini.update_port_s(udp1,udp2)

    my_ini.print()

def test_read_ini():
    print(time.strftime("%d.%m.%y %H:%M:%S"))

    my_ini = MyIni('iu.ini','iu_default.ini')
    print()
    my_ini.print_items(my_ini.config)
    print()

    ips = my_ini.get_ip_s()
    print ('ip s', ips)
    #print ('ip 1', ips[1])

    ports = my_ini.get_port_s()
    print ('ports', ports)

```

Файл «dev.py»

```

import socket
import sys
import time
import struct

from my_utils import sec2hhmmss
from my_utils import int_to_HEX

HEADER_LEN = 4 # msg_code 1 byte, msg_seqn 1 byte, msg_len 2 bytes
MSG_ACK_LEN=4

```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		71

MSG_RUN_BODY_LEN=8+1024 #1032
MSG_RUN_LEN=1032+HEADER_LEN

'''

коды квитанций

'''

ack_tip = '\n'
0 норма\n\
-2 нет связи\n\
0x01 неверный код сообщения\n\
0x11 неверные параметры\n\
0x12 неверный формат (длина) команды\n\
0x13 неверный код команды\n\
0x14 неверный режим задания команды\n\
0x20 команда выполнена с ошибкой\n\
0xEE прибор занят пред.командой\n\
0xFF команда отвергнута'

'''

T -----

'''

'''

- 0x01 - вывод;
- 0x02 - ввод;
- 0x03 -звонок;
- 0x04 - исправить;
- 0x05 - перемотать;
- 0x06 - вернуть зону;
- 0x07 - выдать состояние;
- 0x08 - пропустить зону;
- 0x0A - вернуть группу зону;
- 0x0B - пропустить группу зон;
- 0x0C - диагностика;
- 0x0D - стереть;
- 0x0E - сброс.

'''

comboBox_parm_vnd_t_txt = (
'вывод',
'ввод',
'звонок',
'исправить',
'перемотать',
'вернуть зону',
'выдать состояние',
'пропустить зону',
'вернуть группу зон',
'пропустить группу зон',
'диагностика',

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		72

```

'стереть',
'сброс',
'записать ленточную марку'
)
parm_vnd_t_codes = (0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
0x08, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f)

# извлечь заголовок
def extract_header(reply):
    code_b = (reply)[0:1]
    seq_b = (reply)[1:2]
    len_b = (reply)[2:4]

    seq = int.from_bytes(seq_b, byteorder='little')
    len = int.from_bytes(len_b, byteorder='little')

    # print('code=', code_b, ',', 'seq=', seq, ',', 'body len=',
len)

    return (code_b, seq, len)

# извлечь тело сообщения
def extract_body(reply):
    header_len = 5
    body_pos = header_len - 1
    body_len = len(reply) - header_len + 1
    body = (reply)[body_pos:body_pos+body_len]
    # print('bodylen=', body_len, 'body=', body )

    return (body)

def pack_msg(code, seq, body_len, body):
    fmt1 = '>1B'
    fmt2 = '<1B'
    fmt3 = '<1H'

    if (body_len):
        # print(body_len, type(body_len))
        # print(type(body))
        to_send = struct.pack(fmt1, code) + \
            struct.pack(fmt2, seq) + \
            struct.pack(fmt3, body_len) + \
            body
    else:
        to_send = struct.pack(fmt1, code) + \
            struct.pack(fmt2, seq) + \
            struct.pack(fmt3, body_len)

    # print (to_send, ', len=', len(to_send))
    return to_send

```

Приложение А2. Листинг программы приема отладочных текстовых сообщений.

Файл «udp_log.py»

```
from socket import *
import struct
from my_utils import *

port = 7144

print('регистрация пакетов (порт ',port, ')')
print()
print(str_current_time())
print()

#данные сервера
host = ''
# port = 7142
#host = '192.168.1.83'
addr = (host,port)

udp_socket = socket(AF_INET, SOCK_DGRAM)
#print(udp_socket)

#bind - связывает адрес и порт с сокетом
udp_socket.bind(addr)

while True:

    frame, addr = udp_socket.recvfrom(2048)

    #print('client addr: ', addr)
    tim = str_current_time()
    data = frame
    print(tim)

    lendata = len(data)

    try:
        decoded_str = data.decode()
        print(decoded_str, end='')
    except ValueError:
        print (data)
    #     return False
    else:
        ...
    #     return True
    data = ''

udp_socket.close()
```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		74

Приложение А3. Листинг программы для микроконтроллера.

Файл «et.c»

```
/*
*****
* @file      et.c
*
* @author    Фадеев Е.Д.
*
* @version   V1.0.0
*
*****
*/

#include "main.h"

#ifndef DEBUG_T_OFF
#include "et.h"
#include "log.h"
#include <string.h>
#include "mem.h"

BYTE      *debug_driveT = (BYTE*) DEBUG_DRIVE_MEMORY;
int len_debug_driveT;

BYTE      StartVndTCommand(BYTE      cmd,      BYTE      *status_byte,      BYTE
*diagnostic_byte, const int len, BYTE *data, int* input_len){
    uint8_t  tmp[4];
    LOGPUT("%s cmd=0x%x\n", __FUNCTION__, cmd);

    switch (cmd) {
        case CMD_T_INPUT:
            if (data == 0 || len!=0) return RES_T_ERR_PARAM;
            return et_Read(status_byte,
diagnostic_byte,data,input_len);
        case CMD_T_OUTPUT:
            if ((len < 8 || len > 1024) || (data == 0)) return
RES_T_ERR_PARAM;
            return et_Write(cmd,status_byte, diagnostic_byte,
len, data);
        case CMD_T_WRITE_TM:
            tmp[0] = 0x14;
            return et_Write(cmd,status_byte, diagnostic_byte, 1,
tmp);
        case CMD_T_FIX:
            return et_Write(cmd,status_byte, diagnostic_byte,
len, data);
        case CMD_T_FORWARD_ZONE:
```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		75

```

        return et_Position(cmd, status_byte,
diagnostic_byte, len,data);
        case CMD_T_FORWARD_GROUP:
            return et_Position(cmd, status_byte,
diagnostic_byte, len,data);
        case CMD_T_BACK_ZONE:
            return et_Position(cmd, status_byte,
diagnostic_byte, len,data);
        case CMD_T_BACK_GROUP:
            return et_Position(cmd, status_byte,
diagnostic_byte, len,data);
        case CMD_T_REWIND:
            return et_Position(cmd, status_byte,
diagnostic_byte, len,data);
        case CMD_T_ERASE:
            LOGPUT("CMD_T_ERASE\n");
            return et_ResetEraseRing(cmd, status_byte);
        case CMD_T_RING:
            LOGPUT("CMD_T_RING\n");
            return et_ResetEraseRing(cmd, status_byte);
        case CMD_T_RESET:
            LOGPUT("CMD_T_RESET\n");
            return et_ResetEraseRing(cmd, status_byte);
        case CMD_T_DIAGNOSTICS:
            if (*status_byte != 0x44) return RES_T_ERR_USE;
            return et_Diagnostics(diagnostic_byte);
        case CMD_T_GET_STATUS:
            return et_GetStatus(status_byte);
        default:
            LOGPUT("%s default cmd\n",__FUNCTION__);
////
            return RunCommandT(8,w_Run,7,w_RunWaited);

        return RES_T_UNKNOWN_CMD ; // неверный код команды

    }
    return RES_T_OK;
}

bool t_is_timeout(){
#ifdef DEBUG_BOARD
    return false;
#else
    //TODO
    return false;
#endif
}

BYTE et_ResetEraseRing(BYTE cmd, BYTE *status_byte){ //Reset, Erase,
Ring
    // Выполнить команду (Сброс || Звонок || Стереть)
    fpga_et_write_cmd(cmd);
    // ожидание приема диаг байта

```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		76

```

        while(!fpga_et_is_byte()      &&      !fpga_et_is_timeout()      &&
!t_is_timeout()){};
        if (fpga_et_is_timeout()){
            LOGPUT("%s timeout rcv diag byte\n", __FUNCTION__);
            return RES_T_RUN_TIMEOUT_ERR ;
        }
        *status_byte = fpga_et_read_statusbyte();
#ifdef DEBUG_BOARD
        *status_byte = STATUS_BYTE_READY;
#else
        if (*status_byte != STATUS_BYTE_READY)
            return RES_T_RUN_ERR;
#endif
        LOGPUT("%s: status byte=0x%x \n", __FUNCTION__, *status_byte);
        return RES_T_OK;
    }

BYTE et_Diagnostics(BYTE* diagnostics_byte){
    // Выполнить команду "Диагностика"
    fpga_et_write_cmd(CMD_T_DIAGNOSTICS);
    // ожидание приема диаг байта
    while(!fpga_et_is_byte()      &&      !fpga_et_is_timeout()      &&
!t_is_timeout()){};
    if (fpga_et_is_timeout()){
        LOGPUT("%s timeout rcv diag byte\n", __FUNCTION__);
        return RES_T_RUN_TIMEOUT_ERR ;
    }
    *diagnostics_byte = fpga_et_read_statusbyte();
#ifdef DEBUG_BOARD
    *diagnostics_byte = 0x91;
#else
    LOGPUT("%s      diag      byte      =      0x%02X\n",      __FUNCTION__,
*diagnostics_byte);
    return RES_T_OK;
}

BYTE et_GetStatus(BYTE* status_byte){
    fpga_et_write_cmd(CMD_T_GET_STATUS);

    while(!fpga_et_is_byte()      &&      !fpga_et_is_timeout()      &&
!t_is_timeout()){};
    if (fpga_et_is_timeout()){
        LOGPUT("%s timeout rcv byte\n", __FUNCTION__);
        return RES_T_RUN_TIMEOUT_ERR ;
    }

    *status_byte = fpga_et_read_statusbyte();
#ifdef DEBUG_BOARD
    *status_byte = STATUS_BYTE_CONTINUATION;
#endif
    LOGPUT(" 1 Status byte = 0x%02X\n", *status_byte);
    return 0;
}

```

```

BYTE et_Write(BYTE cmd, BYTE* status_byte, BYTE* diagnostic_byte,
int len, BYTE* data){

    LOGPUT("%s len = %d\n", __FUNCTION__, len);
    LOGPUT("%s data = 0x%02X\n", __FUNCTION__, data[0]);
    len_debug_driveT = len;
    // команда ВЫВОД
    fpga_et_write_cmd(cmd);

    // ожидание приема байта состояния
    while(!fpga_et_is_byte()      &&      !fpga_et_is_timeout()      &&
!t_is_timeout()){};
    if (fpga_et_is_timeout()){
        LOGPUT("%s timeout rcv byte\n", __FUNCTION__);
        return RES_T_RUN_TIMEOUT_ERR ;
    }

    *status_byte = fpga_et_read_statusbyte();
#ifdef DEBUG_BOARD
    *status_byte = STATUS_BYTE_CONTINUATION;
#endif
    LOGPUT(" 1 Status byte = 0x%02X\n", *status_byte);

    // проверка на состояние "Продолжение"
    if(*status_byte != STATUS_BYTE_CONTINUATION){
        LOGPUT("%s sb = 0x%02x is not CONTINUATION  \n",
__FUNCTION__, *status_byte);
        return RES_T_RUN_ERR;
    }

    // Передача массива данных в буфер ПЛИС
    for (int i = 0; i < len; i++){
        #ifdef DEBUG_BOARD
        debug_driveT[i] = data[i];
        #else
        fpga_et_write_fifo(data[i]);
        #endif
    }

    LOGPUT("%s data: (len=%d)\n", __FUNCTION__, len);
    log_dump ((unsigned *) &debug_driveT[0], len);

    // ожидание сигнала вызов
    while(!fpga_et_is_call()      &&      !fpga_et_is_timeout()      &&
!t_is_timeout()){};
    if (fpga_et_is_timeout()){
        LOGPUT("%s timeout rcv call \n", __FUNCTION__);
        return RES_T_RUN_TIMEOUT_ERR ;
    }

    // Команда "Выдать состояние"
    fpga_et_write_cmd(CMD_T_GET_STATUS);

```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		78

```

        // ожидание приема байта состояния
        while(!fpga_et_is_byte()      &&      !fpga_et_is_timeout()      &&
!t_is_timeout()){};
        if (fpga_et_is_timeout()){
            LOGPUT("%s timeout rcv byte\n", __FUNCTION__);
            return RES_T_RUN_TIMEOUT_ERR ;
        }

        *status_byte = fpga_et_read_statusbyte();
#ifdef DEBUG_BOARD
        *status_byte = STATUS_BYTE_READY;
        /*status_byte = STATUS_BYTE_ERROR_BUS;
        /*status_byte = STATUS_BYTE_CMD_RUN_ERROR;
#endif
        LOGPUT(" 2 Status byte = 0x%02X\n", *status_byte);

        // Проверка байта состояния
        if (*status_byte == STATUS_BYTE_READY)
            return RES_T_OK;
        else
            if (*status_byte == STATUS_BYTE_ERROR_BUS)
                return RES_T_RUN_ERR;
            else
                // Команда выполнена с ошибкой
                if (*status_byte == STATUS_BYTE_CMD_RUN_ERROR){
                    // Выполнить команду "Диагностика"
                    fpga_et_write_cmd(CMD_T_DIAGNOSTICS);
                    // ожидание приема диаг байта
                    while(!fpga_et_is_byte()      &&      !fpga_et_is_timeout()      &&
!t_is_timeout()){};
                    if (fpga_et_is_timeout()){
                        LOGPUT("%s timeout rcv diag byte\n", __FUNCTION__);
                        return RES_T_RUN_TIMEOUT_ERR ;
                    }
                    *diagnostic_byte = fpga_et_read_statusbyte();
#ifdef DEBUG_BOARD
                    *diagnostic_byte = 0x97;
#endif
                    LOGPUT("%s  diag  byte  =  0x%02X\n", __FUNCTION__,
*diagnostic_byte);
                    return RES_T_RUN_DIAG_ERR;
                }
            else{
                return RES_T_RUN_ERR;
            }
        }

        BYTE et_Read(BYTE* status_byte, BYTE* diagnostic_byte, BYTE* data,
int* len){
            //int len = 0;

            // Команда "Ввод"

```

```

fpga_et_write_cmd(CMD_T_INPUT);

// ожидание приема байта состояния
while(!fpga_et_is_byte()      &&      !fpga_et_is_timeout()      &&
!t_is_timeout()){};
if (fpga_et_is_timeout()){
    LOGPUT("%s timeout rcv byte\n", __FUNCTION__);
    return RES_T_RUN_TIMEOUT_ERR ;
}

*status_byte = fpga_et_read_statusbyte();
#ifdef DEBUG_BOARD
*status_byte = STATUS_BYTE_BUSY_READY;
#endif
LOGPUT(" 1 Status byte = 0x%02X\n", *status_byte);

// проверка на состояние "Занят, готов"
if(*status_byte != STATUS_BYTE_BUSY_READY){
    LOGPUT("%s sb = 0x%02x is not BUSY_READY \n", __FUNCTION__,
*status_byte);
    return RES_T_RUN_ERR;
}
// ожидание сигнала вызов
while(!fpga_et_is_call()      &&      !fpga_et_is_timeout()      &&
!t_is_timeout()){};
if (fpga_et_is_timeout()){
    LOGPUT("%s timeout rcv call \n", __FUNCTION__);
    return RES_T_RUN_TIMEOUT_ERR ;
}
// Команда "Выдать состояние"
fpga_et_write_cmd(CMD_T_GET_STATUS);

// ожидание приема байта состояния
while(!fpga_et_is_byte()      &&      !fpga_et_is_timeout()      &&
!t_is_timeout()){};
if (fpga_et_is_timeout()){
    LOGPUT("%s timeout rcv byte\n", __FUNCTION__);
    return RES_T_RUN_TIMEOUT_ERR ;
}

*status_byte = fpga_et_read_statusbyte();
#ifdef DEBUG_BOARD
*status_byte = STATUS_BYTE_CONTINUATION;
/*status_byte = STATUS_BYTE_CMD_RUN_ERROR;
#endif
LOGPUT(" 2 Status byte = 0x%02X\n", *status_byte);

// Проверка байта состояния
// Команда выполнена с ошибкой
if (*status_byte == STATUS_BYTE_CMD_RUN_ERROR){
    // Выполнить команду "Диагностика"
    fpga_et_write_cmd(CMD_T_DIAGNOSTICS);
    // ожидание приема диаг байта

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.37.001 ПЗ

Лист

80

```

        while(!fpga_et_is_byte()    &&    !fpga_et_is_timeout()    &&
!t_is_timeout()){};
        if (fpga_et_is_timeout()){
            LOGPUT("%s timeout rcv diag byte\n", __FUNCTION__);
            return RES_T_RUN_TIMEOUT_ERR ;
        }
        *diagnostic_byte = fpga_et_read_statusbyte();
#ifdef DEBUG_BOARD
        *diagnostic_byte = 0x97;
#endif
        LOGPUT("%s    diag    byte    =    0x%02X\n",    __FUNCTION__,
*diagnostic_byte);
        return RES_T_RUN_DIAG_ERR;
    }
    else
    if (*status_byte == STATUS_BYTE_CONTINUATION){

//        // Проверить длину входных данных
//        if (len != fpga_et_read_fifo_cnt())
//            return RES_T_ERR_LEN;

#ifdef DEBUG_BOARD
        *len = 512;
#else
        *len = fpga_et_read_fifo_cnt();
#endif

        // Прием массива данных из буфер ПЛИС
        for (int i = 0; i < *len; i++){
            #ifdef DEBUG_BOARD
                data[i] = debug_driveT[i];
            #else
                data[i] = fpga_et_read_fifo();
            #endif
        }

        LOGPUT("%s data: (len=%d)\n", __FUNCTION__, len);
        log_dump ((unsigned *) &data[0], *len);

        // ожидание сигнала вызов
        while(!fpga_et_is_call()    &&    !fpga_et_is_timeout()    &&
!t_is_timeout()){};
        if (fpga_et_is_timeout()){
            LOGPUT("%s timeout rcv call \n", __FUNCTION__);
            return RES_T_RUN_TIMEOUT_ERR ;
        }
        // Команда "Выдать состояние"
        fpga_et_write_cmd(CMD_T_GET_STATUS);

        // ожидание приема байта состояния
        while(!fpga_et_is_byte()    &&    !fpga_et_is_timeout()    &&
!t_is_timeout()){};
        if (fpga_et_is_timeout()){

```

```

        LOGPUT("%s timeout rcv byte\n", __FUNCTION__);
        return RES_T_RUN_TIMEOUT_ERR ;
    }

    *status_byte = fpga_et_read_statusbyte();
#ifdef DEBUG_BOARD
    *status_byte = STATUS_BYTE_READY;
#endif
    LOGPUT(" 3 Status byte = 0x%02X\n", *status_byte);

    // проверка на состояние "Готов"
    if(*status_byte != STATUS_BYTE_READY){
        LOGPUT("%s sb = 0x%02x is not BUSY_READY \n",
__FUNCTION__, *status_byte);
        return RES_T_RUN_ERR;
    }
    else
        return RES_T_OK;
}
else
    return RES_T_RUN_ERR;
}

BYTE et_Position(BYTE cmd, BYTE* status_byte, BYTE* diagnostic_byte,
int len, BYTE* data)
{
    /* Команда "Пропустить зону" || "Пропустить группу зон" ||
"Вернуть зону" || "Вернуть группу зон" || "Перемотка"*/
    fpga_et_write_cmd(cmd);

    // ожидание приема байта состояния
    while(!fpga_et_is_byte()      &&      !fpga_et_is_timeout()      &&
!t_is_timeout()){};
    if (fpga_et_is_timeout()){
        LOGPUT("%s timeout rcv byte\n", __FUNCTION__);
        return RES_T_RUN_TIMEOUT_ERR ;
    }

    *status_byte = fpga_et_read_statusbyte();
#ifdef DEBUG_BOARD
    *status_byte = STATUS_BYTE_BUSY_READY;
#endif
    LOGPUT(" 1 Status byte = 0x%02X\n", *status_byte);

    // проверка на состояние "Занят, готов"
    if(*status_byte != STATUS_BYTE_BUSY_READY){
        LOGPUT("%s sb = 0x%02x is not BUSY_READY \n", __FUNCTION__,
*status_byte);
        return RES_T_RUN_ERR;
    }
    // ожидание сигнала вызов
    while(!fpga_et_is_call()      &&      !fpga_et_is_timeout()      &&
!t_is_timeout()){};

```

```

    if (fpga_et_is_timeout()){
        LOGPUT("%s timeout rcv call \n", __FUNCTION__);
        return RES_T_RUN_TIMEOUT_ERR ;
    }
    // Команда "Выдать состояние"
    fpga_et_write_cmd(CMD_T_GET_STATUS);

    // ожидание приема байта состояния
    while(!fpga_et_is_byte()      &&      !fpga_et_is_timeout()      &&
!t_is_timeout()){};
    if (fpga_et_is_timeout()){
        LOGPUT("%s timeout rcv byte\n", __FUNCTION__);
        return RES_T_RUN_TIMEOUT_ERR ;
    }

    *status_byte = fpga_et_read_statusbyte();
#ifdef DEBUG_BOARD
    *status_byte = STATUS_BYTE_END_OF_GROUP_ZONE;
#endif

    // Проверка байта состояния
    if (*status_byte == STATUS_BYTE_END_OF_GROUP_ZONE)
        return RES_T_OK;
    else
        return RES_T_RUN_ERR;
}

#endif // #ifndef DEBUG_T_OFF
// EOF

```

Файл «et.h»

```

/*
*****
* @file      et.h
*
* @author    Фадеев Е.Д.
* @version   V1.0.0
*
* @brief     интерфейс eth - T
*
*****
*/

#ifndef __ET_H__
#define __ET_H__

#include "main.h"

// #define CMD_T_WRITE_TM
// ((WORD)0x1001) // Записать ленточную марку

```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		83

```

//#define CMD_T_FORWARD_FILE
      ((WORD)0x1002) // Шаг вперед на файл
//#define CMD_T_BACK_FILE
      ((WORD)0x1003) // Шаг назад на файл

#define CMD_T_OUTPUT
      0x08 // Вывод
#define CMD_T_INPUT
      0x10 // Ввод
#define CMD_T_RING
      0x18 // Звонок
#define CMD_T_FIX
      0x48 // Исправить
#define CMD_T_REWIND
      0x60 // Перемотать
#define CMD_T_BACK_ZONE                                0x78 //
Вернуть зону
#define CMD_T_GET_STATUS
      0x80 // Выдать состояние
#define CMD_T_FORWARD_ZONE                                0x98 //
Пропустить зону
#define CMD_T_BACK_GROUP                                0xA0 //
Вернуть группу зон
#define CMD_T_FORWARD_GROUP                                0xB8 //
Пропустить группу зон
#define CMD_T_DIAGNOSTICS                                0xC0 //
диагностика
#define CMD_T_ERASE
      0xD8 // Стереть
#define CMD_T_RESET
      0xF8 // Сброс

//*****
*****
//Джермейн К.Программирование на IBM360, «Мир»,1978)
#define CMD_T_WRITE_TM
      (0x55) // Записать ленточную марку
//// Под вопросом
//#define CMD_T_FORWARD_FILE
      ((WORD)0x1002) // Шаг вперед на файл
//#define CMD_T_BACK_FILE
      ((WORD)0x1003) // Шаг назад на файл

#define STATUS_BYTE_READY
      0x01 // ГОТОВ
#define STATUS_BYTE_CONTINUATION                                0x02 //
ПРОДОЛЖЕНИЕ
#define STATUS_BYTE_BUSY_READY
      0x03 // ЗАНЯТ, ГОТОВ
#define STATUS_BYTE_INDICATION_REQUEST                                0x82 // УКАЗАНИЕ,
ЗАПРОС
#define STATUS_BYTE_END_OF_GROUP_ZONE                                0x89 // КОНЕЦ
ГРУППЫ ЗОН

```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		84

```

#define STATUS_BYTE_CMD_RUN_ERROR                                0x44  //
КОМАНДА ЗАВЕРШЕНА С ОШИБКОЙ
#define STATUS_BYTE_ERROR_BUS
    0x48  // ОШИБКА ШБ

#define SIGNAL_CALL
    0xEA

// Коды возврата
#define RES_T_OK
    0x00  // норма

#define RES_T_UNKNOWN_CMD
    0xEE  // неверный код команды
#define RES_T_ERR_USE
    0xE1  // неправильное использование команды

// #define RES_T_UNKNOW_ERR
    0xFF
#define RES_T_RUN_ERR
    0xFF  // ошибка при выполнении команды
#define RES_T_RUN_TIMEOUT_ERR
    0xF1  // timeout выполнения команды
#define RES_T_RUN_DIAG_ERR
    0xF2  // ошибка при выполнении команды с диаг байтом
#define RES_T_ERR_PARAM
    0xFD  // ошибка параметров команды
#define RES_T_ERR_LEN
    0xFC  // ошибка ввода : несовпаденит заданной
длины с фактически принятой

```

```

BYTE  StartVndTCommand(BYTE  cmd,  BYTE  *status_byte,  BYTE
*diagnostic_byte, const int len, BYTE *data, int* input_len);
BYTE  et_Diagnostics(BYTE* diag_byte);
BYTE  et_ResetEraseRing(BYTE cmd, BYTE* status_byte); // for Reset,
Erase, Ring (Rer)
BYTE  et_GetStatus(BYTE* staus_byte);
BYTE  et_Write(BYTE cmd, BYTE* status_byte, BYTE* diag_byte, int len,
BYTE* data);
BYTE  et_Read(BYTE* status_byte, BYTE* diagnostic_byte, BYTE* data,
int* len);
BYTE  et_Position(BYTE cmd, BYTE* status_byte, BYTE* diagnostic_byte,
int len, BYTE* data);

#endif  //  __ET_H__

```

Файл «format_process.c»

/*

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		85

```

* @file      format_process.c                                *
* @author    Fadeyev E.D.                                    *
* @version    V1.0.0                                          *
* @brief      управление форматированием - имитация *
*****
*/

#include <MDR1986VE1T.h>
#include "systime.h"
#include "log.h"
#include "format_process.h"

static format_state_t state = FORMAT_START;

static mytime_t format_time_start;
static mytime_t format_time_process;
static mytime_t periodic_test;

#define format_is_working() (GetTime() < format_time_process)

format_state_t format_get_state(void){
    return state;
}

void symulation_format_process(bool* flag_format)
{
    if (*flag_format){
        if(state == FORMAT_START){

            LOGPUT("Start Formatting\n");
            format_time_start = GetTime();
            periodic_test = format_time_start;
            format_time_process = format_time_start +
TIME_TO_FORMAT_SIMULATION;
            state = FORMAT_WORK;
        }
        if (state == FORMAT_WORK){

            if(format_is_working()){
                // debug
                if(isElapsedTime(&periodic_test, 2000)){
                    LOGPUT("Formatting...\n");
                }
            }

            else{
                LOGPUT("Finish Formatting\n");
                *flag_format = false;
            }
        }
    }
}

```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		86

Файл «led.c»

```

/*
*****
* @file      led.c                                *
* @author    Fadeev E.D.                          *
* @version   V1.0.0                                *
* @brief     Debug/Target Board LED Control        *
*****
*/

#include "main.h"

#ifdef USE_LEDS

#include <stdint.h>
#include "MDR32F9Qx_port.h"
#include "MDR32F9Qx_rst_clk.h"
#include "sys_time.h"
#include "led.h"
#include "test.h"
#include "log.h"
#include "fpga.h"

/**
 * @brief  инициализация портов LED
 * @param  void
 * @return void
 */
void LED_Init(void) {
    PORT_InitTypeDef PORT_InitStructure;

    RST_CLK_PCLKcmd(LED_RST_CLK_PCLK, ENABLE);

    PORT_StructInit(&PORT_InitStructure);
    PORT_InitStructure.PORT_Pin    = LED_PORT_Pins; // led.h
    PORT_InitStructure.PORT_OE     = PORT_OE_OUT;
    PORT_InitStructure.PORT_FUNC   = PORT_FUNC_PORT;
    PORT_InitStructure.PORT_MODE   = PORT_MODE_DIGITAL;
    PORT_InitStructure.PORT_SPEED  = PORT_SPEED_SLOW;
    PORT_Init(LED_PORT, &PORT_InitStructure);
}

#define GAP_MS 50
void LED_series(int led, int n, int pulse_time_ms) {
    int i;

    for (i=0 ; i < n; i++) {
        sleep(GAP_MS);
        LED_TOGGLE(led);
        sleep(pulse_time_ms);
        LED_TOGGLE(led);
    };
}

```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		87

```

// -----
//   индикация состояния прибора
// -----
void led_READY(bool on_off){
    #ifdef DEBUG_BOARD
    if (on_off)
        LED_ON(LED_READY);
    else
        LED_OFF(LED_READY);
    #else
        fpga_led_READY(on_off);
    #endif
}

void led_READY_toggle(void){
    static bool state=false;
    state ^= 1;
    #ifdef DEBUG_BOARD
    if (state)
        LED_ON(LED_READY);
    else
        LED_OFF(LED_READY);
    #else
        fpga_led_READY(state);
    #endif
}

void led_ETH_toggle(void){
    static bool state2=false;
    state2 ^= 1;
    if (state2)
        LED_ON(LED_ETH);
    else
        LED_OFF(LED_ETH);
}

void LED_state(void* pvParam){

    for(;;){
        static mytime_t time_ready_to = 0;
        static mytime_t time_ready_to2 = 0;

        if (GetTime() < LED_TIME_STATE_START ) return;

        //Форматирование (от кнопки и по команде из сети)
        if (flag_format){
            //      flag_ready = false;
            if
                (isElapsedTime(&time_ready_to,LED_TIME_FORMAT_PERIOD)){

```

```

//          LOGPUT("LED FMT\n");
//          led_READY_toggle();
//          LED_TOGGLE(LED_MC_WORK);
//      }
//      return;
//  }

// ПРИЕМ
if (eth_reception)
    LED_TOGGLE(LED_MC_ETH_RECEIVE);

// Режим E
if (flag_tcp_work)
    if (flag_tcp_exchange) { // обмен данными с клиентом
        if (isElapsedTime(&time_ready_to, 250))
            led_ETH_toggle();
    }
    else
        LED_ON(LED_ETH);
else
    LED_OFF(LED_ETH); // не режим E

// индикатор РАБОТА МК
if (isElapsedTime(&time_ready_to, LED_TIME_READY_PERIOD))
    LED_TOGGLE(LED_MC_WORK);

// индикатор ГОТОВ (через ПЛИС)
if (flag_ready) {
    led_READY(true);
}
else
    led_READY(false);
}

#endif // USE_LEDS

// EOF

```

Файл «hardware.c»

```

/*
*****
* @file    hardware.c
* @version V1.0.0
* @brief    инициализация оборудования
*****
*/

#include <MDR1986VE1T.h>
#include <stdint.h>

```

```

#include <stdbool.h>
#include <string.h>

// #include "MDR32F9Qx_eeprom.h"
#include "MDR32F9Qx_rst_clk.h"

#include "main.h"
#include "hardware.h"

#include "systime.h"
#include "cfg_eeprom.h"
#include "controls.h"
#include "led.h"

#define DELAY_25MS 25

#define led_pass1() (LED_series(LED_MC_WORK, 3, 20))
#define led_pass2() (LED_series(LED_MC_WORK, 3, 20))
#define led_pass3() (LED_series(LED_MC_WORK, 3, 300))
#define led_err() (LED_series(LED_MC_WORK, 10, 50))
#define led_button_reset_push() (LED_series(LED_MC_WORK, 5, 100))

bool is_restore_cfg = false;
/*-----*/
-----*/
bool hardware_init(void)
{
    mytime_t t=0;
    int res;

//    test_err_code = 0; // норма

//    -----
//    конфигурирование узлов аппаратуры
//    -----

//    органы управления: кнопки и тумблера
    controls_init();
//    органы управления: индикаторы
    LED_Init();

//    тактирование EEPROM
    RST_CLK_PCLKcmd(RST_CLK_PCLK_EEPROM, ENABLE);

//    индикаторы
    LED_ON_ALL();
    sleep(500);
    LED_OFF_ALL();

// LED_OFF_ALL();
// индикация прохождения стартового внутреннего теста
    led_pass1();

```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		90

```

// // время начала инициализации (смотреть в shell)
// t0_mcu_init = GetTime();

// опрос кнопки СБРОС
if (controls_button_reset()) {
    led_button_reset_push();
    is_restore_cfg = true;
}

res=1;
// индикация прохождения теста
if (res)
    LED_series(LED_MC_WORK,10,25);

// -----
// восстановление настроек в EEROM
// -----

// запись в EEPROM
if (is_restore_cfg) {
    __NOP();
    cfg_restore();
}
return(res);
}

/*-----
-----*/

```

Файл «utils.c»

```

/*
*****
* @file    utils.c                      *
* @version V1.0.0                      *
* @brief   вспомогательные функции    *
*****
*/

#include <stdio.h>
#include <time.h>
#include <stdint.h>
#include <string.h>

#include "utils.h"
#include "sys_time.h"
#include "net.h"
#include "log.h"

```

```

/*-----*/
-----*/
int time2string(void *buf, time_t time){
    struct tm * timeinfo;
    //char buffer [20];
    int sz;
    timeinfo = localtime (&time);
    sz = strftime (buf,40,"%d.%m.%y %H:%M:%S",timeinfo);
    return(sz);
}
static char sEplapseTime[20] = "15:46:00";
char* getsElapsedTime(void) {
    uint32_t now = GetTime()/1000;
    int sz = sec2hhmmss(sEplapseTime,now);
    sEplapseTime[sz]=0;
    return (sEplapseTime);
}
/*-----*/
-----*/
int sec2hhmmss(void *buf, uint32_t t){
    int tim = (int)(t);
    int hh = (tim /3600);
    int mm = (int)((tim - hh*3600)/60);
    int ss = tim - hh*3600 - mm*60;
    sprintf(buf,9,"%02d:%02d:%02d", hh, mm, ss);
    return(8);
}
/*-----*/
-----*/
/*
преобразование строки в массив , содержащий MAC
возвращает
    1 - норма,
    0 - ошибка
*/
int str2mac(char *str, eth_addr_t *mac){
    uint32_t m[6];
    int n;

    if (strlen(str) != 17) return 0;

    // n = sscanf (str, "%2x:%2x:%2x:%2x:%2x:%2x", &mac->addr[0],
    // &mac->addr[1],&mac->addr[2],&mac->addr[3],&mac->
    // >addr[4],&mac->addr[5]);

    n = sscanf (str, "%2x:%2x:%2x:%2x:%2x:%2x", &m[0], &m[1],
    &m[2], &m[3], &m[4], &m[5]);
    mac->addr[0] = (uint8_t) m[0];
    mac->addr[1] = (uint8_t) m[1];
    mac->addr[2] = (uint8_t) m[2];
    mac->addr[3] = (uint8_t) m[3];
    mac->addr[4] = (uint8_t) m[4];

```

```

        mac->addr[5] = (uint8_t) m[5];

        if (n!=6)            return 0;
        return 1;
    }
    /*-----*/
    /*
преобразование массива, содержащий MAC в строку
*/
void mac2str(char *str, eth_addr_t *mac){
    snprintf(str, 20, "%02x:%02x:%02x:%02x:%02x:%02x",
        mac->addr[0], mac->addr[1], mac->addr[2],
        mac->addr[3], mac->addr[4], mac->addr[5]);
}
    /*-----*/
    /*
#include "inet.h"
*/
преобразование строки с IP-адресом в IP-адрес
с контролем входных данных
retcode : 0 - ошибка
                                1 - норма
*/
int str2ip(char *str, uip_ipaddr_t *ip){
    uip_ipaddr_t    tmp_ip;

    // правильный адрес содержит 3 точки: "192.168.1.35"
    {
        int cnt=0, i;
        for (i = 0; i < strlen(str); i++) if (str[i] == '.') cnt++;
        if (cnt != 3)
            return 0;
    }

    if (inet_aton(str, (struct in_addr *) &tmp_ip) != 1) return 0;
    *ip = tmp_ip;
    return 1;
}
    /*-----*/
    /*
преобразование IP-адреса в строку
*/
void ip2str(uip_ipaddr_t ip_addr, char *s_ip){
    int sz;
    sz
    snprintf(s_ip, 17, "%d.%d.%d.%d", uip_ipaddr_to_quad(&ip_addr));
    s_ip[sz]=0;
}

```

Файл «systime.c»

```

/*
*****
* @file      systime.c
* @version   V1.0.0
* @brief     функции для работы со временем
*****
*/

// systime.c

#include "MDR1986VE1T.h"
#include "main.h"
#include "systime.h"

mytime_t GetElapsedTime(mytime_t time)// посчитать кол-во прошедшего
времени
{
    mytime_t tmp = 0, st;
    //st = SystemTime;
    st = GetTime();
    if(time <= st)
        return st - time;
    else
    {
        tmp = ~tmp;
        tmp = tmp - time;
        return tmp + st;
    }
}

void sleep(mytime_t time)
{
    mytime_t t;
    t=GetTime();
    while(GetElapsedTime(t)<time)
    {
        __NOP();
    }
    return;
}

/**
* @brief     проверка, истекло ли время (ms) с момента предыдущей
проверки
* @param     адрес где хранится ожидаемое время
* @param     период проверки
* @return    true - время истекло, false - нет
*/
bool isElapsedTime(mytime_t *tim_to, mytime_t period){
    mytime_t curr = GetTime();

```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		94

```

        if (curr > *tim_to){
            *tim_to = curr + period;
            return(true);
        }
        else
            return(false);
    }

/**
 * @brief  инициализация системных часов
 * @param  void
 * @return void
 */
void SysTimer_Init(void){
    #if (defined (SYSTIMER_SYSTICK))
        SysTick_lms();
    #elif defined (SYSTIMER_TIMER1)
        TIMER1_Init();
    #else
        #error "System Timer is not defined"
    #endif
}
// EOF

```

Файл «net.c»

```

/*
*****
 * @file      net.c
 * @version   V1.0.0
 * @brief     The net TCP/IP stack based on uip
*****
*/

#include <MDR1986VE1T.h>
#include <stdint.h>
#include <stdbool.h>
#include <string.h>

#include "MDR32F9Qx_eeprom.h"
#include "MDR32F9Qx_rst_clk.h"
#include "MDR32F9Qx_eth.h"

#include "main.h"
#include "net.h"
#include "uip.h"
#include "cfg_eeprom.h"
#include "eeprom_api.h"
// #include "controls.h"

/*-----*/
-----*/

```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		95

```

bool Enable_HSE2(void );
void eth_init(void);
void fun_tcpinit(void);
void fun_udpinit(void);

/*-----*/
-----*/
extern cfg_t cfg;
//net_macaddr_t my_mac;

eth_addr_t my_mac;

//net_macaddr_t rev_my_mac;
net_ip4addr_t my_ip ;
//net_ip4addr_t rev_my_ip;

net_ip4addr_t my_router ;
net_ip4addr_t my_netmask ;

//uint16_t      my_port = 0; // на 18.08.16 не исп, 12.10.16 - исп
в UDPRcv, udp_send,
uint16_t      my_port_udp = 0; // 26.01.17
uint16_t      my_port_tcp_ctrl = 0;
uint16_t      my_port_tcp_info = 0;

net_ip4addr_t * net_get_ip(void){
    return( (net_ip4addr_t *) &my_ip);
}
void net_cfg_set(void){
//  my_mac.u16[0]      = cfg.mac[0];
//  my_mac.u16[1]      = cfg.mac[1];
//  my_mac.u16[2]      = cfg.mac[2];

    my_mac.addr[0]      = cfg.mac[0];
    my_mac.addr[1]      = cfg.mac[1];
    my_mac.addr[2]      = cfg.mac[2];
    my_mac.addr[3]      = cfg.mac[3];
    my_mac.addr[4]      = cfg.mac[4];
    my_mac.addr[5]      = cfg.mac[5];

    //my_ip.u32          = cfg.u32;
    memcpy((void*)&my_ip, (void *) &cfg.ip, 4);
    memcpy((void*)&my_router, (void *) &cfg.router, 4); // 17.08.17
    memcpy((void*)&my_netmask, (void *) &cfg.netmask, 4); //
17.08.17

    my_port_udp          = cfg.port_udp;
    my_port_tcp_info     = cfg.port_tcp_info;
    my_port_tcp_ctrl     = cfg.port_tcp_ctrl;
}
bool net_init(void){
    uint8_t net_addr_3minor_bits;

```

```

bool res;

#ifdef DEBUG_CFG_SET_DEFAULT
cfg_set_default();
#else // из EEPROM
cfg_read_eeprom_and_set();
cfg_restore(); // установка заводских настроек - разовая
отладочная
#endif

// конфигурирование и запуск работы сетевого интерфейса
#ifdef DEBUG_ETH_OFF
#warning "DEBUG_ETH_OFF"
#else
#warning "ETH ACTIVE"
net_cfg_set();

// тактирование eth
res = Enable_HSE2();
if (res == false) return(false); // нет тактирования - выход по
ошибке

MDR_RST_CLK->ETH_CLOCK = (3 << 28) | (1 << 27) | (1 << 24);
// Enable Peripheral Clock

// fun_tcpinit();
eth_init(); // инициализация eth и uip
return(true);
#endif

}
// EOF

```

Файл «log.c»

```

/*
*****
* @file      log.c
* @version   V1.0.0
* @brief     журнал UDP send broadcast
*****
*/
#include "main.h"

#ifdef LOG

#include "log.h"
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>

#include "sys_time.h"

```

```

#include "utils.h"
#include "udp_uip.h"

#include <FreeRTOS.h>
#include <queue.h>
#include <task.h>
#include <semphr.h>

#
static char* log_buf;

static char receive_buf[LOGFIFOSIZE] = {0};

#define LOGSIZE() (strlen(log_buf))

#define LOGGET() (_log_buf)

void init_log_fifo(void){
    if (log_buf){
        xQueueLog = xQueueCreate(Queue_LENGTH, Queue_ITEM_SIZE);
        if (xQueueLog == NULL){
            //
        }
    }
}

static int prefix(char* buffer){
    int64_t now = 0;
    int sz = 0;

    now = GetTime();

    buffer[0] = '#';
    sprintf(&buffer[6], "%05lld", now);
    strcat(buffer, " ");

    sz = strlen(buffer);
    return sz;
}

static int vlog_put(char *format,va_list par);

void log_put(char *txt,...){

    static char* buffer;
    static char sendBuf[256];
    //static char buf[100];
    va_list par;
    va_start(par, txt);

    //strcpy(buffer, txt);
    //memcpy(buf, &txt, strlen(txt));

```

```

        //sprintf(buffer, "%d", sz);

        vsprintf(sendBuf,txt,par);
        buffer = sendBuf;
        //buf = sendBuf;
        //memcpy(allbuf, receive_buf, sz_str);

        xQueueSend(xQueueLog,&buffer,portMAX_DELAY); // работает
        //xQueueSend(xQueueLog, &sendBuf, portMAX_DELAY);
    }
char* vlog_put(const char *format,va_list par){

    int sz_str = 0;

    char allbuf[LOGRECSIZE];

    int len_pref = 0;
    int i;

    sz_str = vsnprintf(receive_buf,LOGRECSIZE-80,format,par);
    memcpy(allbuf, receive_buf, sz_str);

    return allbuf;

}

void vLogFlushTask(void* pvParam){

    int symbCount = 0;
    int i;
    char* recvBuf;

    for(;;){

        if (uxQueueMessagesWaiting(xQueueLog) != 0) {}

        xQueueReceive(xQueueLog, &recvBuf, portMAX_DELAY);

        udp_send_broadcast(recvBuf,strlen(recvBuf));

        memset(&recvBuf, 0, sizeof(recvBuf));
        //vTaskDelay(1000);
    }
}

/*-----*/
-----*/
/**
 * @brief Вывод в журнал дампа памяти
 * @param addr - начальный адрес
 * @param size - размер области <= LOGDUMP_ITEMS_MAX)
 * @retval void

```

```

*/

void log_dump (unsigned * addr, int size)
{
    unsigned *p = addr;
    unsigned *pend = p + size/sizeof(unsigned int);
    unsigned i, j;
    // unsigned char *buf = my_buf2;
    int buf_index =0;
    int sz;
    bool work = true;
    bool is_truncated=false;

    if (size > LOGDUMP_ITEMS_MAX) {
        size = LOGDUMP_ITEMS_MAX;
        pend = p + size/sizeof(unsigned int);
        is_truncated=true;
        LOGPUT("[%x,%x]                                     sz=%d
%s\n",p,pend,size,is_truncated?"TRUNCATED!":"");
    }

    LOGPUT("[%x,%x]                                     sz=%d
%s",p,pend,size,is_truncated?"TRUNCATED!":"");

    while (work) {
        //LOGPUT("\n %08X: ", p);
        LOGPUT("\n");
        for (j = 0; j < 8; j++) {
            LOGPUT(" %08X", *p);
            p++;
            if (p >= pend) {
                LOGPUT("\n\n");
                work = false;
                break;
            }
        }
    }

}

/*-----*/
-----*/
#endif // LOG

```

Файл «firmware.c»

```

/*
*****
* @file    firmware.c                                *
* @version V1.0.0                                    *
* @brief    работа с описаниями прошивки            *

```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		100

```

*****
*/

#include "MDR1986VE1T.h"
#include "system_MDR1986VE1T.h"
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <ctype.h>

#include "main.h"
#include "mem.h"
#include "timestamp.h"
#include "firmware_defs.h"
#include "utils.h"

#define FW_ID  (unsigned int)(0x33303030)

int32_t firmware_id = FW_ID;
uint32_t firmware_timestamp = TIMESTAMP;

// рабочие ячейки
uint32_t firmware_crc    = 0x1234567;
uint32_t firmware_size   = 45678;

// для передачи по сети
firmware_info_TypeDef firmware_info;

void firmware_info_set(void){
    firmware_info.id    = firmware_id;
    firmware_info.crc    = firmware_crc;
    firmware_info.timestamp = firmware_timestamp;
    firmware_info.size = firmware_size;
}

static char _buf[64]={0};

char * getsFirmware(void)
{
    char ts_string[80];
    uint32_t ts = firmware_info.timestamp + MSK_HOURS * 3600;
    int sz;

    sz = time2string(ts_string,ts);
    ts_string[sz] = 0;

    sz = snprintf((char *) _buf, sizeof(_buf) ,
//          "%s id=0x%08X crc=0x%08X size=%d",
//    ts_string,          firmware_info.id,          firmware_info.crc,
firmware_info.size);
    "%s crc=0x%08X size=%d",
    ts_string, firmware_info.crc, firmware_info.size);
}

```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		101

```

        _buf[sz]=0;

        return _buf;
    }

    /*-----*/
    -----*/
int getLenFirmware() {
    return strlen(_buf);
}

// EOF

```

Файл «call_uip.c»

```

/*
*****
* @file      call_uip.c                      *
* @version   V1.0.0                          *
* @brief     инициализация и вызов uip      *
*****
*/

/* Based on uip (2001) by Adam Dunkels.
 * ...
 * This file is part of the uIP TCP/IP stack.
 *
 */

#include "main.h"

#include "appselect.h"
#include "typedefs.h"
#include "uip.h"
#include "uip_arp.h"
#include "tapdev.h"
#include "config.h"
#include "timer.h"

#include "net.h"
#include "tcpserver.h"

#define BUF ((struct uip_eth_hdr *)&uip_buf[0])

#ifndef NULL
#define NULL (void *)0
#endif /* NULL */

static struct uip_eth_addr default_ethaddr;
static uint8_t my_ip_default[4];
static uint8_t my_router_default[4];
static uint8_t my_netmask_default[4];

```

```

static uip_ipaddr_t ipaddr;
struct timer    periodic_timer, arp_timer;

int is_link, is_crs  = 0;
/*-----*/
-----*/
bool eth_init(void)
{
    // чтение конфигурации, установленной в net_cfg()
    net_ipaddr_copy((void*)&my_ip_default, (void*)&my_ip);
    net_ipaddr_copy((void*)&my_router_default, (void*)&my_router); //
17.08.17
    net_ipaddr_copy((void*)&my_netmask_default, (void*)&my_netmask);
// 17.08.17

    net_macaddr_copy(&default_ethaddr, &my_mac);

    uip_setethaddr(default_ethaddr);

    //timer_set(&periodic_timer, CLOCK_CONF_SECOND / 2); // orig
uip example
    timer_set(&periodic_timer, CLOCK_CONF_SECOND / 100); // 10 ms
    // пропуск пакетов при задержке теста компьютера меньше 2 раз
времени таймера

    timer_set(&arp_timer, CLOCK_CONF_SECOND * 10);
    tapdev_init((unsigned char*) default_ethaddr.addr);

    uip_init();
    uip_arp_init();

    uip_sethostaddr(&my_ip_default);

    uip_setdraddr(&my_router_default);

    uip_setnetmask(&my_netmask_default);

#ifdef    TCP_SERVER
{
        tcpserver_init();
}
#endif    //TCP_SERVER1

    return(true);
}

#include <FreeRTOS.h>
#include <task.h>
#include <semphr.h>

void vCallUipTask_uIP_periodic(void *pvParameters) {
    uint32_t i;

```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		103

```

uint8_t delay_arp = 0;

for (;;) {

    vTaskDelay(50); // попробовать уменьшить

    delay_arp++;
#ifdef UIP_TCP
    for (i = 0; i < UIP_CONNS; i++) {
        uip_periodic(i);
        if (uip_len > 0) {
            uip_arp_out();
            tapdev_send((uint8_t *) uip_buf, uip_len);
        }
    }
#endif //UIP_TCP

#ifdef UIP_UDP
    for(i = 0; i < UIP_UDP_CONNS; i++) {
        uip_udp_periodic(i);
        if(uip_len > 0) {
            uip_arp_out();
            tapdev_send(uip_buf,uip_len);
        }
    }
#endif /* UIP_UDP */

    if(timer_expired(&arp_timer)){
        timer_reset(&arp_timer);
        uip_arp_timer();
    }
}

void vCallUipTask_uIP(void *pvParameters) {

    for (;;) {
        uip_len = tapdev_read((uint8_t *) uip_buf);
        //
        if (uip_len > 0) {
            xSemaphoreGive(xSemaphore_EthPacketReceived);
            //vTaskDelay(10);
            if (BUF->type == htons(UIP_ETHTYPE_IP)) {
                uip_arp_ipin();
                uip_input();
                if (uip_len > 0) {
                    uip_arp_out();

                    tapdev_send((uint8_t *) uip_buf,
uip_len);
                }
            }
            else if (BUF->type == htons(UIP_ETHTYPE_ARP)) {

```

```

        uip_arp_arpin();
        if (uip_len > 0) {

            tapdev_send((uint8_t *) uip_buf,
uip_len);

        }

    }

    //taskYIELD();
    vTaskDelay(5); // так работает стабильнее, чем taskYIELD()
}

/*-----*/
-----*/

// EOF

```

Файл «udp_uip.c»

```

/*

*****
*****
* @file      udp_uip.c
* @version V1.0.0
* @brief     UDP для вызова uip
*             *****
*****
*/

#include <string.h>
#include <stdarg.h>

#include "main.h"
#include "uip.h"
#include "uip_arp.h"
#include "tapdev.h"

#define DEBUG_UDP

#ifdef DEBUG_UDP
#include "udp_uip.h"

/*-----*/
-----*/
void udp_send(void *data, int length) {
    #if (UIP_UDP == 1)
        send_data_buf(data, length);
    #endif
}

```

```

/*-----
-----*/
#ifdef UIP_UDP == 1
void udp_send_conn(void *data, const int len, struct uip_udp_conn
*conn)
{
    extern uint16_t uip_slen;
    // extern void *uip_sappdata;
    uip_udp_conn = conn;
    uip_slen = len;

    //memcpy(uip_appdata, data, uip_slen); // ???
    memcpy(&uip_buf[UIP_LLH_LEN + UIP_IPUDPH_LEN], data, len >
UIP_BUFSIZE ? UIP_BUFSIZE: len);
    uip_process(UIP_UDP_SEND_CONN);
    uip_arp_out();
    tapdev_send(uip_buf, uip_len);
    uip_len = 0;
}

/*-----
-----*/

static struct uip_udp_conn *broadcast_conn;
void
udp_broadcast_init(void)
{
    uip_ipaddr_t addr;
    uip_ipaddr_t myip = get_ip_cfg();
    //uint16_t port = BROADCAST_PORT + myip.u8[3];
    uint16_t port = BROADCAST_PORT;

    //uip_ipaddr_t *dest = {255,255,255,255};
    #ifdef LOG_BROADCAST
    uip_ipaddr(&addr, 255,255,255,255); // так работает тоже
    #else
    uip_ipaddr(&addr, 192,168,101,208);
    #endif

    broadcast_conn = uip_udp_new(&addr, HTONS(port));
    if(broadcast_conn != NULL) {
        uip_udp_bind(broadcast_conn, HTONS(port));
    }
}

/*-----
-----*/
void udp_send_broadcast(void *data, int length) {
    udp_send_conn(data, length, broadcast_conn);
}

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.37.001 ПЗ

Лист

106

```

/* -----
-----*/

#endif    //(UIP_UDP == 1)

#endif // DEBUG_UDP

// EOF

    Файл «controls.c»

/*
*****
* @file      controls.c                      *
* @version  V1.0.0                          *
* @brief     органы управления прибором      *
*****
*/

#include "main.h"

#ifdef CONTROLS_ON

#include "MDR1986VE1T.h"
#include "system_MDR1986VE1T.h"
#include "MDR32F9Qx_rst_clk.h"
#include "MDR32F9Qx_port.h"

#include "controls.h"
#include "led.h"

#ifdef DEBUG_BOARD
uint16_t  controls_button(uint16_t pin){

    uint16_t res=0;
    int i;
    res =0;
    for (i = 0; i < 5; i++)
        res = res | !(BUTTON_CONTROL_read(pin));
    return res;
}
#endif // DEBUG_BOARD

uint16_t  controls_port_read(void){
    return PORT_ReadInputData(BTNS_PORT);
}

#define DEBUG_SWITCH

```

```

#define DEBUG_SWITCH_BITS 0

uint8_t controls_get_eth_addr_3bit(void){
    uint8_t bits = DEBUG_SWITCH_BITS;
    #ifndef DEBUG_SWITCH
    bits = ADDR_ETH0() + (ADDR_ETH1() << 1) + (ADDR_ETH2() << 2);
    #endif
    return(bits);
}

void controls_init(void){
    PORT_InitTypeDef PORT_InitStructure;

    RST_CLK_PCLKcmd(CONTROLS_BTNS_RST_CLK_PORT, ENABLE);

    PORT_InitStructure.PORT_FUNC = PORT_FUNC_PORT;
    PORT_InitStructure.PORT_MODE = PORT_MODE_DIGITAL;
    PORT_InitStructure.PORT_SPEED = PORT_SPEED_FAST;

    #ifdef DEBUG_BOARD
    PORT_InitStructure.PORT_OE = PORT_OE_IN;
    PORT_InitStructure.PORT_Pin = (BUTTON_RESET_PIN +
    BUTTON_FORMAT_PIN + BUTTON_MODE_PIN);
    PORT_InitStructure.PORT_PULL_UP = PORT_PULL_UP_ON;
    #else
    PORT_InitStructure.PORT_Pin = (BUTTON_MODE_PIN);
    // PORT_InitStructure.PORT_PULL_UP = PORT_PULL_UP_ON;
    // PORT_InitStructure.PORT_PULL_DOWN = PORT_PULL_DOWN_ON;
    #endif
    PORT_InitStructure.PORT_SPEED = PORT_SPEED_SLOW;
    PORT_Init(BTNS_PORT, &PORT_InitStructure);
}

#endif //CONTROLS_ON

// EOF

```

Файл «main.c»

```

/*
*****
* @file      main.c
* @version   V1.0.0
* @brief     diplom
*****
*/

/*
Based on uip (2001) by Adam Dunkels.
*/

```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		108

```

#include <MDR1986VE1T.h>
#include <stdint.h>
#include <stdbool.h>
#include <string.h>

#include "MDR32F9Qx_eeprom.h"
#include "MDR32F9Qx_rst_clk.h"

#include "main.h"
#include "timestamp.h"

#include "sys_time.h"
#include "firmware_defs.h"
#include "eeprom_api.h"
#include "net.h"
#include "udp_uip.h"

#include "log.h"
#include "led.h"
#include "controls.h"
#include "device/format_process.h"

#include "tcpserver.h"

#include "dev.h"

#include "device\et.h"

//---- FreeRTOS
#include <FreeRTOS.h>
#include <task.h>
#include <queue.h>
#include <semphr.h>
#include <timers.h>

// прототипы
void CPU_init_Q8MHz ( void );
void CPU_init_Q12MHz ( void );
void SystemCoreClockUpdate2 (void);

// -----
// прототипы задач
// -----
void vHardwareMcuReadyTask(void* pvParam);
void vCallUipTask_uIP_periodic(void* pvParameters);
void vCallUipTask_uIP(void* pvParameters);
void vLedStateTask(void* pvParameters);
void vLogFlushTask(void* pvParameters);
void vWorkWithDev(void* pvParameters);
void vLedState(void* pvParameters);

// -----

```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		109

```

// семафоры и мьютексы
// -----
SemaphoreHandle_t xSemaphore_EthPacketReceived;
volatile SemaphoreHandle_t xMutexAllowSend;
SemaphoreHandle_t xSemaphore_TcpExch;
SemaphoreHandle_t xSemaphore_Error;
SemaphoreHandle_t xSemaphore_FormatProc;
SemaphoreHandle_t xSemaphore_Work;

// -----
// очереди
// -----
QueueHandle_t xQueueLog;
QueueHandle_t xQueueOut;
QueueHandle_t xQueueIn;

// -----
// таймеры FreeRTOS (периодически и интервальные)
// -----
xTimerHandle      xOneShotTimers;
xTimerHandle xTestTimer;
xTimerHandle xStatusTimer;

// -----
// приоритеты задач
// -----
#define mainReady_MCU_TASK_PRIORITY
(tskIDLE_PRIORITY+8)
#define mainLOG_TASK_PRIORITY
(tskIDLE_PRIORITY+6)
#define mainLED_STATE_TASK_PRIORITY
(tskIDLE_PRIORITY+4)
#define mainUIP_PERIODIC_TASK_PRIORITY
(tskIDLE_PRIORITY+3)
#define mainUIP_TASK_PRIORITY
(tskIDLE_PRIORITY+2)
#define mainLOG_FLUSH_TASK_PRIORITY
(tskIDLE_PRIORITY+1)
#define mainDEV_WORK_TASK_PRIORITY
(tskIDLE_PRIORITY+7)

// -----
// параметры очередей
// -----
#define mainLOG_QUEUE_LENGTH      128
#define mainLOG_QUEUE_ITEM_SIZE   sizeof(char*)

#define mainOUT_QUEUE_LENGTH      256
#define mainOUT_QUEUE_ITEM_SIZE   sizeof(unsigned char*)

#define mainIN_QUEUE_LENGTH      256
#define mainIN_QUEUE_ITEM_SIZE   sizeof(unsigned char*)

```

```

void vTimersFunction(xTimerHandle xTimer){
    log_put("UIP_APPDATA_SIZE : %d\n", UIP_APPDATA_SIZE);
}

static void eth_dispatcher(void){
    xTaskCreate(vCallUipTask_uIP_periodic,                (const
char*)"UipPeriodicTask",
    configMINIMAL_STACK_SIZE*2,                        NULL,
mainUIP_PERIODIC_TASK_PRIORITY, (xTaskHandle*) NULL);

    xTaskCreate(vCallUipTask_uIP, (const char*) "UipTask",
    configMINIMAL_STACK_SIZE*2, NULL, mainUIP_TASK_PRIORITY,
(xTaskHandle*) NULL);
}

void log_info(bool t, bool is_restore_cfg){
    extern int d_minor_bits;

    // информация о прошивке в лог
    LOGINIT();
    LOGPUT("start\n");
    LOGPUT("\n");
    LOGPUT("%s\n", getsFirmware());
//

    LOGPUT("CPUCLK %d MHz\n", CPUCLK);
    LOGPUT("ADDR[0..2]=%d\n", d_minor_bits);
    LOGPUT("IP=%s MAC=%s\n", getsIP(), getsMAC());

    services_log();
    LOGPUT("UDP SERVER PORT: %d\n", *get_port_udp());

    #ifdef TCP_SERVER
    LOGPUT("TCP SERVER 1 PORT: %d\n", *get_port_tcp_ctrl());
    #endif

    LOGPUT("T = %d\n", t);
    LOGPUT("restore net cfg = %d\n", is_restore_cfg);
}

extern bool power_on_T;

int main(void)
{
    // -----
    // конфигурирование аппаратуры
    // -----

    // тактирование и запуск системных часов
    #if HSE_VALUE==8000000
    CPU_init_Q8MHz();

```

```

#endif
#if HSE_VALUE==12000000
CPU_init_Q12MHz();
#else
#endif

SysTimer_Init();

// стартовый тест внутренней аппаратуры (IROM1, IRAM1, IRAM2 )
if (!test_internal_start()) while(1) {};

// инициализация узлов МК
hardware_init();

// инициализация сетевого интерфейса и TCP/IP
is_eth = net_init();
udp_broadcast_init();

// информация о прошивке
firmware_info_set();

// информация о прошивке в лог
log_info(power_on_T, is_restore_cfg);

flag_err = false;

LOGPUT(delimiter);
is_dev_ready = dev_start(power_on_T);
mytime_t time_to_start_dev = GetTime()+TIME_DELAY_SINCE_
START;

// семафор получения пакета (ICMP протокол)
xSemaphore_EthPacketReceived = xSemaphoreCreateBinary();

// семафор работы по TCP
xSemaphore_TcpExchg = xSemaphoreCreateBinary();

// создание мьютексов
xMutexAllowSend = xSemaphoreCreateMutex();

// очередь для лога
xQueueLog = xQueueCreate(mainLOG_QUEUE_LENGTH,
mainLOG_QUEUE_ITEM_SIZE);

// очереди для хранения сообщений передав. По TCP
xQueueIn = xQueueCreate(mainIN_QUEUE_LENGTH,
mainIN_QUEUE_ITEM_SIZE);
xQueueOut = xQueueCreate(mainOUT_QUEUE_LENGTH,
mainOUT_QUEUE_ITEM_SIZE);

```

```

        // создание таймеров
        xTimer = xTimerCreate("OneShotTimer", 12000 / portTICK_RATE_MS,
pdFALSE, NULL, vTimersFunction);

        xTimerReset(xOneShotTimers, 0);

        eth_dispatcher();

        xTaskCreate(vTestLog_Task, (const char*) "TestLog",
            configMINIMAL_STACK_SIZE*2, NULL, mainLOG_TASK_PRIORITY,
(xTaskHandle*) NULL);

        xTaskCreate(vHardwareMcuReadyTask, (const char*)
"McuReadyTask",
            configMINIMAL_STACK_SIZE*2, (void*) power_on_switch_M2T,
mainReady_MCU_TASK_PRIORITY, (xTaskHandle*) NULL);

        xTaskCreate(vLedStateTask, (const char*) "LedStateTask",
            configMINIMAL_STACK_SIZE*2, NULL,
mainLED_STATE_TASK_PRIORITY, (xTaskHandle*) NULL);

        xTaskCreate(vLogFlushTask, (const char*) "Flush Task",
            configMINIMAL_STACK_SIZE*2, NULL,
mainLOG_FLUSH_TASK_PRIORITY, (xTaskHandle*) NULL);

        xTaskCreate(vDevWork, (const char*) "DEV Task",
            configMINIMAL_STACK_SIZE*2, NULL,
mainDEV_WORK_TASK_PRIORITY, (xTaskHandle*) NULL);
        xTaskCreate(vLedState, (const char*) "LedState Task",
            configMINIMAL_STACK_SIZE*2, NULL,
mainLED_STATE_TASK_PRIORITY, (xTaskHandle*) NULL);

        vTaskStartScheduler();

        for( ;; );
    }

    #if 1
    // This FreeRTOS callback function gets called once per tick (default
    = 1000Hz).
    // -----
    void vApplicationTickHook( void ) {
        ++u64Ticks;
    }

    // This FreeRTOS call-back function gets when no other task is ready
    to execute.
    // On a completely unloaded system this is getting called at over
    2.5MHz!
    // -----
    void vApplicationIdleHook( void ) {

```

```

        ++u64IdleTicksCnt;
    }

// A required FreeRTOS function.
// -----
void vApplicationMallocFailedHook( void ) {
    configASSERT( 0 ); // Latch on any failure / error.
}

```

Файл «main.h»

```

/*
*****
* @file      main.h
* @version   V1.0.0
* @brief     настройки ПО МК
*****
*/

#ifndef __MAIN_H
#define __MAIN_H

#define SYSTIMER_TIMER1

#define DEBUG_BOARD_VER_6 // for ver. 6

#define DEBUG_APPLICATION_OFF

// #define DEBUG_T_OFF

// -----

#define USE_LEDS // индикаторы всегда используются

#define CONTROLS_ON // подключить ф-ции органов управления прибором

// -----
// подключение протоколов
// -----

#ifdef TCP_SERVER
#define TCP_SERVER_APP // подключение пользовательских функций
tcp_server_app.c

#define TCP_SERVER1 // подключение сервера 1

// #define TCP_RECONNECT_CHECK // проверка установления нового
соединения после закрытия предыдущего

#endif

```

```

// -----
// сетевые настройки
// -----

#ifdef IG_EQ_1
#define MAC0 0x01
#else
#define MAC0 0x02
#endif
#define MAC1 0x02
#define MAC2 0x03
#define MAC3 0x04

// #define MAC4 0x01
// #define MAC5 0x90
#define MAC4 0x05
#define MAC5 0x60 // ? 0x06 ?

#define IP_0 192
#define IP_1 168
#define IP_2 101
#define IP_3 91

// -----
// порты серверов
// -----
#define TCP_PORT1 7139

#define LOG

// -----
#define LOG_BROADCAST
#define BROADCAST_PORT 7144

#define LOGRECSIZE (1400)

#endif // UDP_KERNEL_SENDER

#define TIME_DEV_PERIOD (5*1000)
#define TIME_DEV_TIMEOUT_START_T (55*1000)

// #define TIME_DEV_TIMEOUT_START_T (1*1000)
#define TIME_DEV_FORMAT (12*60*1000)

// #define TIME_TEST_PERIOD (30*1000)

#define TIME_LOG_MARK 3*60*1000

#define MYBUFSIZE 1024

// -----
// типы данных

```

					ПГУ 2.090301.37.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		115

```

// -----

#include <stdint.h>
#include <stdbool.h>

#define vu32 volatile uint32_t
#define vu8  volatile uint8_t
#define u32  uint32_t
#define u16  uint16_t
#define u8   uint8_t

#define u32_t uint32_t
#define u16_t uint16_t
#define u8_t  uint8_t

#define DWORD      unsigned int
#define WORD       unsigned short
#define BYTE       unsigned char

// #define uchar    unsigned char

#include <MDR1986VE1T.h>
#define htons(n)  (__REV16(n))
#define htonl(n)  (__REV(n))

bool is_app_mode(void); // режим работы прикладного уровня

extern bool flag_restart;
#define reload() (flag_restart=true)

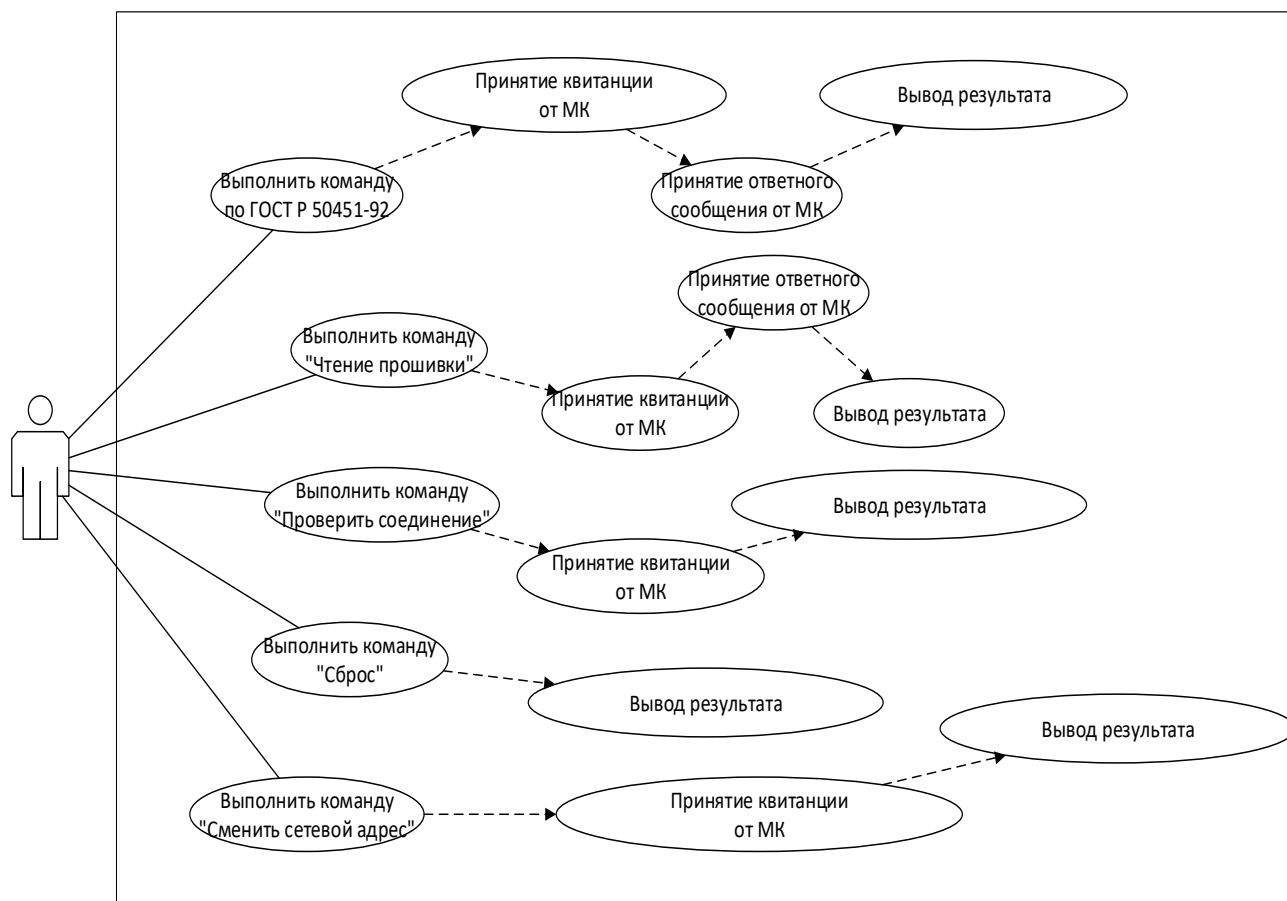
#endif

// EOF

```

Приложение В. Графическая часть

					ПГУ 2.090301.37.001 ПЗ	Лист
						117
Изм.	Лист	№ докум.	Подпись	Дата		



ПГУ 2.090301.37.003 Д1

Изм.	Лист	№ докум.	Подпись	Дата
Разраб.	Фадеев Е.Д.			
Провер.	Коннов Н.Н.			
Н. Контр.	Кучин А.В.			
Утверд.	Митрохин М.А.			

Технологическое программное обеспечение микроконтроллера платы дисковых интерфейсов.
Программа управления.
Диаграмма вариантов использования.

Лит.	Лист	Листов
		1
ФВТ, каф. ВТ гр. 16ВВ1		



Нажать на кнопку "Сброс"

Перезагрузка микроконтроллера

Нажать на кнопку "Форматирование"

Форматирование накопителя

ПГУ 2.090301.37.003 Д2

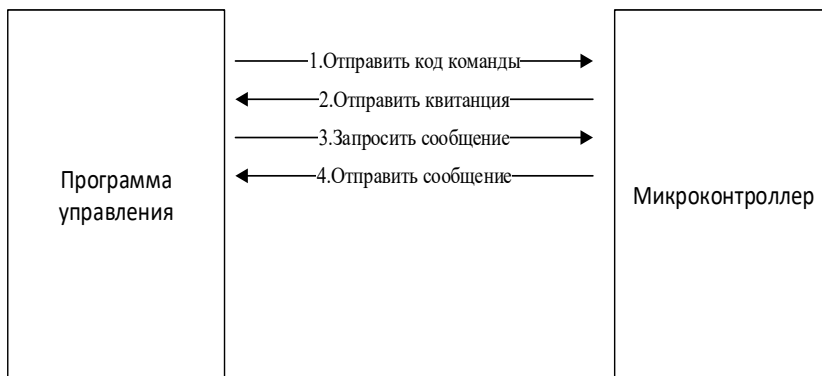
Изм.	Лист	№ докум.	Подпись	Дата
Разраб.		Фадеев Е.Д.		
Провер.		Коннов Н.Н.		
Н. Контр.		Кучин А.В.		
Утверд.		Митрохин М.А.		

Технологическое программное
обеспечение микроконтроллера
платы дисковых интерфейсов.
Программа для микроконтроллера.
Диаграмма вариантов использования.

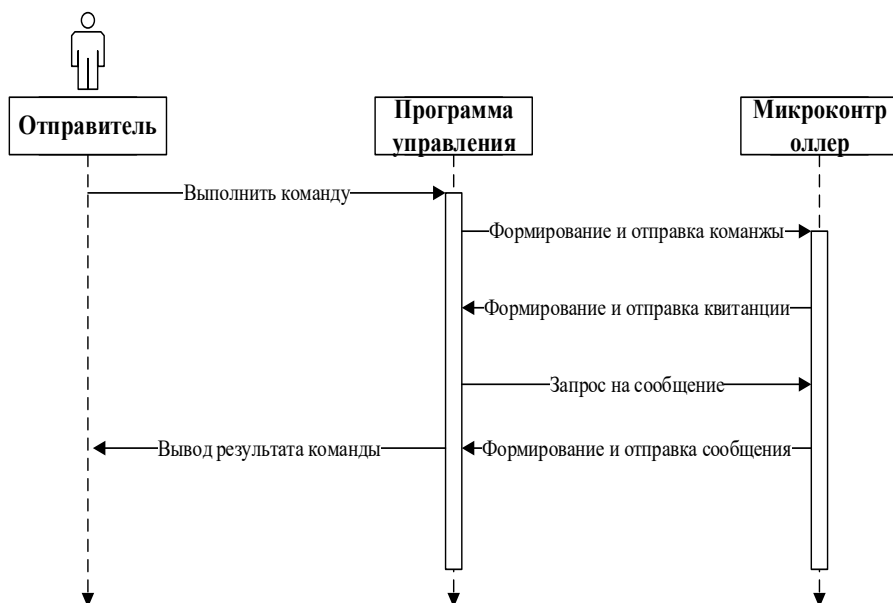
Лит. Лист Листов

1

ФВТ, каф. ВТ
гр. 16ВВ1



					ПГУ 2.090301.37.003 ДЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Фадеев Е.Д.			Технологическое программное обеспечение микроконтроллера платы дисковых интерфейсов. Диаграмм коммуникации.	Лит.	Лист
Провер.		Коннов Н.Н.					Листов
Н. Контр.		Кучин А.В.					1
						ФВТ, каф. ВТ гр. 16ВВ1	
Утверд.		Митрохин М.А.					



ПГУ 2.090301.37.003 Д4

					ПГУ 2.090301.37.003 Д4		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.	Фадеев Е.Д.				Технологическое программное обеспечение микроконтроллера платы дисковых интерфейсов. Диаграмм последовательности.	Лит.	Лист
Провер.	Коннов Н.Н.						Листов
Н. Контр.	Кучин А.В.						1
						ФВТ, каф. ВТ гр. 16ВВ1	
Утверд.	Митрохин М.А.						

Приложение С. Презентация

Выпускная квалификационная работа «Технологическое программное обеспечение микроконтроллера платы дисковых интерфейсов»

Автор работы: студент группы 16вв1 Фадеев Е.Д.

Научный руководитель: к.т.н., профессор каф. ВТ ПГУ Коннов Н.Н.

Постановка задачи

Требуется разработать технологическое системное программное обеспечение (далее ТПО), предназначенное для отладки работы со специализированным накопителем. ТПО представляет комплекс программ, состоящий из программы управления, расположенное на ПЭВМ, текстовых отладочных сообщений, расположенной на ПЭВМ и программы для микроконтроллера.

- ▶ Программа управления предназначена для передачи общих команд взаимодействия с микроконтроллером, а также команд взаимодействия с накопителем, описанных в ГОСТ Р 50451-92, приема ответных сообщений с результатами выполнения команд.
- ▶ Программа приема отладочных сообщений предназначена для приема сообщений, выдаваемых во время отладки программы для микроконтроллера.
- ▶ Программа для микроконтроллера предназначена для эмуляции пространства дискового накопителя в ОЗУ микроконтроллера, размеров в 4 Кб, приема команд от программы управления, выполнение команд общего назначения, трансляцию команд по ГОСТ Р 50451-92 на эмулятор пространства дискового накопителя, формирования результатов выполнения команд и отправку их программе управления.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.37.001 ПЗ

Лист

122

Выбор микроконтроллера

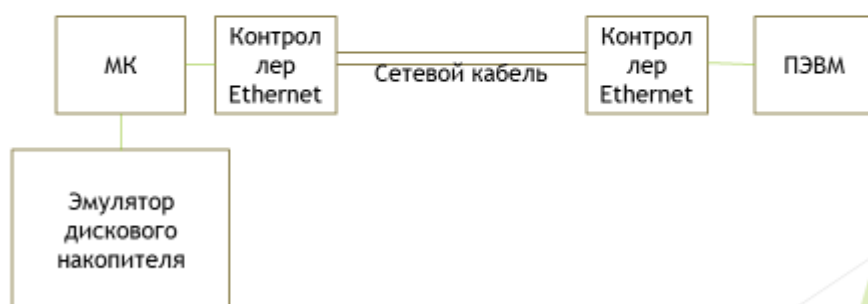
В качестве микроконтроллера был выбран 32-разрядный микроконтроллер для авиационного применения 1986BE1T от компании «Миландр», поставляемый вместе с отладочной платой.

Основными плюсами данного микроконтроллера для выполнения ВКР явились:

- ▶ Наличие цифрового интерфейса Ethernet со скоростью передачи 10/100 (потребуется для организации сопряжения с ПЭВМ)
- ▶ Высокие показатели объема flash-памяти программ (128 Кбайт) и ОЗУ (48 Кбайт). (программа для микроконтроллера будет исполняться по управлению операционной системы реального времени FreeRTOS, что потребует наличие дополнительного объема flash-памяти программ и на базе микроконтроллер будет развернут tcp-сервер, что подразумевает под собой наличие TCP/IP стека, который будет занимать дополнительный объем flash-памяти программ).



Общая схема сопряжения микроконтроллера с ПЭВМ



Программа управления. Стек технологий

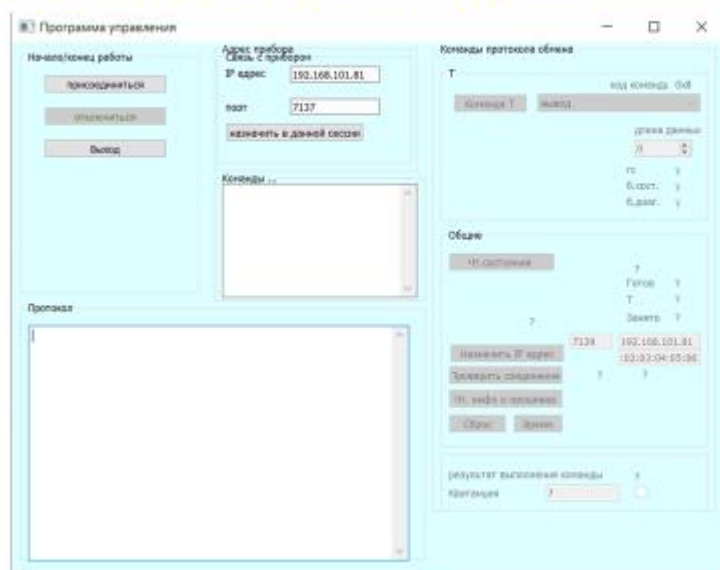
Программа управления выполняется на ПЭВМ под управлением операционной системы Windows. Служит для отправки в микроконтроллер команд управления и приема ответных сообщений от микроконтроллера.

Написана на языке программирования Python версии 3.7 с использованием фреймворка PyQt версии 5.0. Python был выбран в качестве языка программирования, так как разработка на нем значительно быстрее, чем например на языках C/C++, Java. Однако за скорость написания приходится платить производительностью программ, но т.к. в ВКР основное внимание уделялось написанию программы для микроконтроллера - этот язык отлично подошел.

В качестве фреймворка для написания графического интерфейса послужил PyQt. Он обладает большим набором графических виджетов, удобным механизмом коммуникации между объектами сигнал-слот. И, например в отличие от библиотеки для создания графических интерфейсов tkinter (входящей в состав стандартной библиотеки Python) поставляется с редактором готовых форм Qt Designer.



Главное окно программы управления



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.37.001 ПЗ

Лист

124

Обмен между программой управления и микроконтроллером

Алгоритм обмена со стороны программы управления представлен ниже в виде псевдокода:

```
call: выбор команды
call: упаковка команды
call: отправка упакованной команды
call: прием квитанции
IF квитанция=0 THEN
    call: прием сообщения
    call: разбор сообщения
ELSE
    DISPLAY ошибка формирования команды
ENDIF
DISPLAY квитанция
DISPLAY сообщение
```

Алгоритм обмена со стороны программы для микроконтроллера представлена ниже в виде псевдокода:

```
call: прием упакованной команды
call: разбор упакованной команды
call: выполнение команды
call: формирование квитанции
call: отправка сформированной квитанции
IF квитанция=0 THEN
    call: формирование сообщения с результатом
    call: отправка сообщения с результатом
ELSE
    call: ошибка в квитанции
ENDIF
```

Структура команды, передаваемой от программы управления в микроконтроллер

Id	seq	Body_len	Body
----	-----	----------	------

Поле Id - идентификатор команды. Далее описаны коды идентификаторов и соответствующие команды: «Прочитать состояние» (0x01), «Проверить соединение» (0x02), «Выполнить команду Т» (0x03), «Чтение прошивки» (0x04), «Чтение времени» (0x05), «Смена сетевого адреса» (0x06), «Сброс» (0x07)

Поле seq - счетчик поступающих от команду управления команд.

Поле body_len - длина, передаваемых данных (во всех командах кроме «Выполнить команду Т» поле = 0)

Поле body - передаваемые данные (во всех командах кроме «Выполнить команду Т» поле = 0)

Программа для приема текстовых отладочных широкоэмительных сообщений. Стек технологий.

Программа для приема текстовых отладочных широкоэмительных сообщений представляет собой UDP сервер.

Написана на языке программирования Python версии 3.7. Единственное, что выполняет программа - это прослушивание порта и вывод, полученной отладочной информации в консоль. Язык Python явился идеальным решением для реализации программы, т.к. он особо удобен для написания небольших скриптов.

Для того, чтобы отправить отладочную информацию на сервер, разработчику программного обеспечения микроконтроллеров необходимо вызвать функцию `log_put(const char*...)` в теле своей программе. Эта функция повторяет поведение функции `printf(const char*...)` из стандартной библиотеки Си, с единственным исключением - она выводит данные в буфер, который потом отправляется по протоколу UDP на программу для приема отладочных сообщений.

Программа для микроконтроллера. Стек технологий.

- ▶ Для написания программы для микроконтроллера был выбран язык программирования. Является одним из самых популярных языков программирования встраиваемых систем. В число его плюсов входит: простота изучения, скорость разработки и т.д. Например если сравнить С другим не менее популярным языком разработки встраиваемых систем язык С будет проигрывать по производительности, однако разработка на нем куда приятнее чем на Ассемблере. Код - читабельнее, дальнейшая поддержка проектов проще.
- ▶ Программа для микроконтроллера будет исполняться на операционной системе реального времени FreeRTOS версии 9.0. В плюс использования ОСРВ можно выделить многозадачность, приоритеты задач, инструменты межзадачного взаимодействия(очереди, семафоры, мьютексы). FreeRTOS - является одной из самых популярных ОСРВ на сегодняшний день. Плюсы: бесплатная, портирована на большое количество железа, мощный функционал.
- ▶ Для организации развертывания tcp сервера на микроконтроллера был выбран tcp/ip стек uIP. Основным его плюсом является то, что он занимает мало места и в нем реализовано небольшое количество протоколов стека tcp/ip, а т.к. нам нужен лишь tcp и udp протоколы - он является идеальным для нас.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.37.001 ПЗ

Лист

126

- ▶ В программе для микроконтроллера будет реализована эмуляция команд взаимодействия с накопителем по ГОСТ Р 50451-92. В качестве команд будут выступать команды «Выдать состояние», «Диагностика», «Вывод», «Ввод», «Звонок», «Стереть», «Сброс». Пространство накопителя будет эмитировать ОЗУ микроконтроллера. И так как ОЗУ микроконтроллера лишено недостатков накопителя. Поведение работы команд необходимо будет задавать программно.
- ▶ Команда «Вывод» выполняет запись в накопитель.
- ▶ Команда «Ввод» выполняет чтение из накопителя
- ▶ Команда «Сброс», «Звонок», «Стереть» выдает байт состояния «Готов»
- ▶ Команда «Выдать состояние» выдает последний сформированный байт состояния
- ▶ Команда «Диагностика» выдает диагностический байт состояния.

- ▶ Команда «Назначить Ip - адрес»

```

> ping -n -s 32 192.168.101.91
Узел пакеты => 192.168.101.91 по < 32 байтам даны:
Пакет от 192.168.101.91: число байт=32 время=1мс TTL=128
Пакет от 192.168.101.91: число байт=32 время=1мс TTL=128
Пакет от 192.168.101.91: число байт=32 время=1мс TTL=128
Пакет от 192.168.101.91: число байт=32 время=1мс TTL=128
Пакет от 192.168.101.91: число байт=32 время=1мс TTL=128

Статистика Ping до 192.168.101.91:
  Пакетов: отправлено = 4, получено = 4, потеряно = 0
  (0% потерь)

Наибольшее время приема-передачи в мс:
  Максимальное = 0 мс, Максимальное < 1 мс, Среднее < 0 мс

> tcp_get len=16 opcode=0x20, seq=0 bodylen=12
CMD SET if ADDR ip=192.168.101.92 mac=02:02:03:04:05:60 port=7139
cfg_save done
< ack send opcode=0x0
< dev form msg 0xAA msg body len=4

```

```

#show ipconfig 192.168.300.50

Узел системы с 192.168.100.50 по шк 32 Address:
Имя от 192.168.100.50: Значение уже существует.
Получено сетевое соединение для термина.
Получено сетевое соединение для термина.
Получено сетевое соединение для термина.

#show ipconfig 192.168.300.50

        Базовый: состояние = 4, получено = 1, потеряно = 1
(70K байт)

#show ipconfig 192.168.300.50

Узел системы с 192.168.100.50 по шк 32 Address:
Имя от 192.168.100.50: число Rcv-Idl превышает 711-120
Имя от 192.168.100.50: число Rcv-Idl превышает 711-120
Имя от 192.168.100.50: число Rcv-Idl превышает 711-120
Имя от 192.168.100.50: число Rcv-Idl превышает 711-120

#show ipconfig 192.168.300.50

        Базовый: состояние = 4, получено = 4, потеряно = 0
(0K байт).

Состояние системы можно увидеть в см.
содержащемся в меню "Настройка" с 1 сек. Состояние = 0

```

► Команда «Считать время МК»

```
21.06.20 05:42:43.877
> tcp_get len=4 opcode=0x12 , seq=2 bodylen=0
< ack_send code=0x0
< dev_form_msg 0xAA msg_body_len=4
21.06.20 05:42:44.267
uptime=00:05:26 (326) s
< dev_form_msg 0x12 msg_body_len=4
```



► Команда «Вывод»

```
21.06.20 17:26:35.924
> tcp_get len=76 opcode=0x6 , seq=4 bodylen=72
< ack_send code=0x0
< dev_form_msg 0xAA msg_body_len=4
21.06.20 17:26:36.417
CMD_T_H2M msg_body:
[20000470,200004f0] sz=128
00000000 00000040 A4A3A2A1 A5A7A6A5 ACABA9A9 B0AFAEAD B4B3B2B1 B0B7B6B5
BCB9BAB9 C0BFBEED C4C3C2C1 C8C7C6C5 CCCBCAC9 D0CFCECD D4D3D2D1 D8D7D6D5
DCB8DAD9 E0DFDEED 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Start Process
Drive_T_Start cmd=0x06
StartTCommand cmd=0x0
et_Write len = 44
et_Write data = 0xA1
1 Status byte = 0x02
et_Write data: (len=44)
[20103000,20103040] sz=64
A4A3A2A1 A5A7A6A5 ACABA9A9 B0AFAEAD B4B3B2B1 B0B7B6B5 BCB9BAB9 C0BFBEED
C4C3C2C1 C8C7C6C5 CCCBCAC9 D0CFCECD D4D3D2D1 D8D7D6D5 DCB8DAD9 E0DFDEED

2 Status byte = 0x01
Drive_T_Start setcode=0x00 stat_byte=0x01 diag_byte=0x00
Finish Process
Drive_T_Termination rc=0x00000000
< msg_send_res rc=0x00000100, data len=0
< msg_send_res opcode=0x05 body_len=1032
[20000450,200004e0] sz=128
00000405 00000100 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Протокол

21.06.20 17:26:36 : операция обращения 0x0: норма

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.37.001 ПЗ

Лист

128

► Команда «Ввод»

```
21.04.20 17:28:46.132
> tcp_get len=12 opcode=0x6 , seq=5 bodylen=8
< tcp_send code=0x0
< dev_firm_msg 0x00 msg_body_len=4
21.04.20 17:28:46.547
CMD_T_WHM msg_body:
[20000a70,20000a50] sz=128
00000010 00000040 A4A3A2A1 A5A7A6A5 ACANAAA9 B0AF7EAD D4D3D2D1 D0D7D6D5
BCBBBA99 C0FBFBED C4C3C2C1 C0C7C6C5 C0CBAC99 D0CFCECD D4D3D2D1 D0D7D6D5
D0CBAD99 E0DFDEED
00000020 00000050 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

START PROCESS
Drive_T_Start cmd=0x10
StartCommand cmd=0x10
1 Status byte = 0x01
2 Status byte = 0x02
at_read data: (len=64)
[20100900,20100940] sz=64
A4A3A2A1 A5A7A6A5 ACANAAA9 B0AF7EAD D4D3D2D1 D0D7D6D5 D0CBAD99 C0FBFBED
C4C3C2C1 C0C7C6C5 C0CBAC99 D0CFCECD D4D3D2D1 D0D7D6D5 D0CBAD99 E0DFDEED

3 Status byte = 0x01
Drive_T_Start retcode=0x00 stat_byte=0x01 diag_byte=0x00
Finish Process
Drive_T_Termination rc=0x00000000
< msg_send_res rc=0x00000010, data len=64
< msg_send_res opcode=0x05 body_len=1032
[20000450,20000460] sz=128
00400505 00000100 00000040 A4A3A2A1 A5A7A6A5 ACANAAA9 B0AF7EAD D4D3D2D1
B0B7B6B5 BCBBBA99 C0FBFBED C4C3C2C1 C0C7C6C5 C0CBAC99 D0CFCECD D4D3D2D1
D0D7D6D5 D0CBAD99 E0DFDEED 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Протокол

```
21.04.20 17:28:46 : операция обращения 0x10: норма
21.04.20 17:28:46 : считать данные
21.04.20 17:28:46 : -----
00000000 A4A3A2A1 A5A7A6A5 ACANAAA9 B0AF7EAD
00000010 B4B3B2B1 B0B7B6B5 BCBBBA99 C0FBFBED
00000020 C4C3C2C1 C0C7C6C5 C0CBAC99 D0CFCECD
00000030 D4D3D2D1 D0D7D6D5 D0CBAD99 E0DFDEED
```

Заключение

- В ходе выполнения дипломной работы был разработан комплекс программ для отладки работы с специализированным накопителем. Данный проект будет использоваться, как шаблон при дальнейшей разработки реальной платы, входящей в состав специализированного устройства.

Спасибо за внимание!

					ПГУ 2.090301.37.001 ПЗ	Лист
						130
Изм.	Лист	№ докум.	Подпись	Дата		