

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«Южно-Уральский Государственный Университет  
(национальный исследовательский университет)»  
филиал в г. Миассе  
факультет «Электротехнический»  
кафедра Автоматики  
Специальность 27.03.04 «Управление в технических системах»

ДОПУСТИТЬ К ЗАЩИТЕ  
Заведующий кафедрой

\_\_\_\_\_  
подпись / ФИО  
« \_\_\_\_ » \_\_\_\_\_ 2023 г.

Система исполнения для программируемого логического контроллера  
(утверждено Приказом №746-13/12 от 20.04.2023 г.)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ  
ЮУрГУ-27.03.04.2023.203.23.05 ВКР

Автор работы  
студент группы МиЭт-523

\_\_\_\_\_  
подпись / ФИО  
« \_\_\_\_ » \_\_\_\_\_ 2023 г.

Руководитель работы

\_\_\_\_\_  
должность  
\_\_\_\_\_  
подпись / ФИО  
« \_\_\_\_ » \_\_\_\_\_ 2023 г.

Консультант

\_\_\_\_\_  
должность  
\_\_\_\_\_  
подпись / ФИО  
« \_\_\_\_ » \_\_\_\_\_ 2023 г.

Нормоконтроль

\_\_\_\_\_  
должность  
\_\_\_\_\_  
подпись / ФИО  
« \_\_\_\_ » \_\_\_\_\_ 2023 г.

Миасс 2023

а) базироваться на операционной системе реального времени;

б) обеспечить работу каналов дискретного ввода в режимах: выключен, нормальный дискретный, счетчик импульсов, тахометр, одно- и двухканальный счетный

энкодер;

в) обеспечить работу каналов дискретного вывода в режимах: выключен, нормальный дискретный, ШИМ, безопасный;

г) обеспечить работу каналов аналогового ввода в режимах: выключен, нормальный аналоговый (также выполнить опрос встроенного датчика температуры);

д) обеспечить работу управляющей программы, реализованной на языках стандарта МЭК-61131-3 в открытой и свободно распространяемой среде программирования Veremiz;

е) обеспечить работу первого интерфейса RS485 для дистанционной отладки управляющей программы из среды Veremiz в режиме реального времени;

ё) обеспечить работу второго интерфейса RS485 для дистанционного доступа к регистрам данных «ПЛК411» по протоколу MODBUS RTU (режим «Slave»);

ж) обеспечить работу светодиодов (USER, RUN, ERR, NET).

3.4 При формировании исходного кода предусмотреть возможность его портирования на другие платформы: разделить код на аппаратнозависимый, аппаратно-независимый общий и уровня операционной системы.

3.5 В разработке должны использоваться открытые и доступные решения.

4 Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов)

#### 4.1 ОБЗОРНЫЙ РАЗДЕЛ

##### 4.1.1 Введение в стандартный ПЛК

###### 4.1.1.1 Назначение и структура контроллера

###### 4.1.1.2 Классификация контроллеров

###### 4.1.1.3 Встраиваемое программное обеспечение

###### 4.1.1.4 Базовая логика работы контроллера

##### 4.1.2 Операционная система реального времени

###### 4.1.2.1 Многозадачность

###### 4.1.2.2 Планировщик

###### 4.1.2.3 События и прерывания

###### 4.1.2.4 Межпроцессное взаимодействие и общие ресурсы

###### 4.1.2.5 Программные таймеры

##### 4.1.3 Стандарт МЭК-61131-3

##### 4.1.4 Среда разработки Veremiz

###### 4.1.4.1 Схема создания и загрузки управляющей программы

###### 4.1.4.2 Связь управляющей программы и системы исполнения

#### 4.2 ОСНОВНОЙ РАЗДЕЛ

##### 4.2.1 Функциональная схема «ПЛК411»

##### 4.2.2 Схема многозадачности системы исполнения

##### 4.2.3 Карта регистров данных

##### 4.2.4 Разработка системы исполнения

##### 4.2.5 Создание описания целевой платформы

5 Перечень графического материала (с точным указанием обязательных чертежей, плакатов, слайдов)

##### 5.1 Функциональная схема «ПЛК411»

##### 5.2 Схема многозадачности системы исполнения «ПЛК411»

6 Консультанты по работе (с указанием относящихся к ним разделов работы)

[illegible]

## 7 Дата выдачи задания

Руководитель \_\_\_\_\_ / \_\_\_\_\_  
подпись ФИО

Консультант \_\_\_\_\_ / \_\_\_\_\_  
подпись ФИО

Задание принял к исполнению \_\_\_\_\_ / \_\_\_\_\_  
подпись студента ФИО

## 8 Календарный план

Наименование этапов выпускной квалификационной работы	Срок выполнения этапов работы	Отметка о выполнении руководителя
Формирование технического задания	10.03.2023	
Обзор классов программируемых устройств	13.03.2023	
Обзор встраиваемого программного обеспечения устройств управления	14.03.2023	
Обзор стандарта МЭК-61131-3	15.03.2023	
Обзор среды разработки Beremiz	17.03.2023	
Анализ и разработка функциональной схемы	22.03.2023	
Анализ и разработка схемы многозадачности системы исполнения	24.03.2023	
Анализ и разработка карты адресов регистров данных	28.03.2023	
Анализ и выбор средств разработки системы исполнения	29.03.2023	
Разработка системы исполнения: конфигурация системы, адресное пространство	03.04.2023	
Разработка системы исполнения: каналы дискретного ввода (все режимы)	07.04.2023	
Разработка системы исполнения: каналы дискретного вывода (все режимы)	11.04.2023	
Разработка системы исполнения: каналы аналогового ввода (все режимы)	14.04.2023	
Разработка системы исполнения: исполнение управляющей программы	21.04.2023	
Разработка системы исполнения: протокол отладки	26.04.2023	
Разработка системы исполнения: протокол MODBUS	05.05.2023	
Отладка и тестирование системы исполнения	10.05.2023	

Заведующий кафедрой \_\_\_\_\_ /  
подпись \_\_\_\_\_ ФИО \_\_\_\_\_

Руководитель работы \_\_\_\_\_ /  
подпись \_\_\_\_\_ ФИО \_\_\_\_\_

Консультант \_\_\_\_\_ /  
подпись \_\_\_\_\_ ФИО \_\_\_\_\_

Студент \_\_\_\_\_ /  
подпись \_\_\_\_\_ ФИО \_\_\_\_\_

## АННОТАЦИЯ

Звездин В.В. Система исполнения для программируемого логического контроллера. – Миасс: ЮУрГУ, Автоматика; 2023, 124 с., 27 таблиц, 30 иллюстраций, библиогр. список - 35 наим., 3 прил., 1 лист схемы формата А1, 1 лист схемы формата А2.

После анализа актуальных проблем отечественного рынка промышленной автоматике и основываясь на практическом опыте производителей программируемых устройств управления был предложен вариант программируемого контроллера на базе доступных и открытых программных решений.

Для программируемого контроллера с аппаратным ядром архитектуры ARM Cortex-M4 разработана встраиваемая система исполнения, которая обеспечивает работу периферийных устройств контроллера (каналы ввода/вывода, сетевые интерфейсы) в требуемых режимах и выполняет пользовательскую управляющую программу, реализованную на высокоуровневых языках стандарта МЭК-61131-3 в открытой и свободно распространяемой среде программирования Veremiz. Сама система исполнения контроллера также реализована с использованием доступных и открытых средств: язык программирования Си, среда программирования Eclipse, операционная система реального времени FreeRTOS, кросс-компилятор GNU ARM GCC.

					27.03.04.2023.203.23.05 ВКР		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.					ПЛК411. Система исполнения	Лит.	Лист
Провер.							Листов
Конс.							2
Н. Контр.						ЮУрГУ (НИУ) Кафедра Автоматики	
Утв.							
							124

## ОГЛАВЛЕНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ.....	7
ВВЕДЕНИЕ.....	13
1 ОБЗОРНЫЙ РАЗДЕЛ.....	15
1.1 Введение в стандартный ПЛК.....	15
1.1.1 Назначение и структура.....	15
1.1.2 Классификация контроллеров.....	17
1.1.2.1 Интеллектуальные реле.....	18
1.1.2.2 Моноблочные контроллеры.....	18
1.1.2.3 Модульные контроллеры.....	19
1.1.2.4 Распределенные системы.....	22
1.1.3 Встраиваемое программное обеспечение.....	24
1.1.3.1 Система исполнения.....	24
1.1.3.2 Управляющая программа.....	25
1.1.4 Базовая логика работы.....	26
1.2 Операционная система реального времени.....	28
1.2.1 Многозадачность.....	30
1.2.2 Планировщик.....	31
1.2.3 События и прерывания.....	33
1.2.4 Межпроцессное взаимодействие и общие ресурсы.....	34
1.2.4.1 Очередь.....	34
1.2.4.2 Семафор.....	35
1.2.4.3 Мьютекс.....	36
1.2.4.4 Критические секции.....	36
1.2.5 Программные таймеры.....	37
1.3 Стандрат МЭК-61131-3.....	37
1.3.1 Язык ST.....	38
1.3.2 Язык IL.....	39
1.3.3 Язык LD.....	39
1.3.4 Язык FBD.....	40
1.3.5 Язык SFC.....	40
1.3.6 Среда программирования.....	41
1.4 Среда программирования Beremiz.....	43

1.4.1	Veremiz YAPLC.....	45
1.4.2	Схема создания и загрузки управляющей программы.....	46
1.4.3	Связь системы исполнения и управляющей программы.....	49
1.4.4	Связь переменных с регистрами данных.....	51
2	ОСНОВНОЙ РАЗДЕЛ.....	54
2.1	Функциональная схема «ПЛК411».....	54
2.1.1	Технические характеристики ПЛК.....	54
2.1.2	Технические характеристики микроконтроллера.....	57
2.1.3	Распределение памяти.....	58
2.1.4	Режим работы ПЛК.....	58
2.1.4.1	Режим «Перезагрузка».....	59
2.1.4.2	Режим «Обновление встроенного программного обеспечения»...	59
2.1.4.3	Режим «Работа».....	59
2.1.5	Каналы «DI».....	60
2.1.5.1	Общие настройки.....	60
2.1.5.2	Режим «Выключен».....	60
2.1.5.3	Режим «Нормальный».....	60
2.1.5.4	Режим «Счетчик импульсов».....	61
2.1.5.5	Режим «Тахометр».....	62
2.1.5.6	Режим «Инкрементальный энкодер (счетчик)».....	63
2.1.5.7	Режим «Инкрементальный энкодер (счетчик и тахометр)».....	64
2.1.5.8	Сброс счетчиков.....	65
2.1.6	Каналы «DO».....	65
2.1.6.1	Общие настройки.....	65
2.1.6.2	Режим «Выключен».....	65
2.1.6.3	Режим «Нормальный».....	65
2.1.6.4	Режим «Быстрый».....	66
2.1.6.5	Режим «ШИМ».....	66
2.1.7	Каналы «AI».....	67
2.1.7.1	Общие настройки.....	68
2.1.7.2	Режим «Выключен».....	68
2.1.7.3	Режим «Нормальный».....	68
2.1.8	Безопасное состояние каналов вывода.....	68
2.1.9	Сторожевой таймер.....	70



2.1.10	Системная информация.....	71
2.1.11	Системные команды и настройки.....	72
2.1.12	Пользовательские регистры данных.....	73
2.1.13	Интерфейс «COM.1».....	73
2.1.14	Интерфейс «COM.2».....	75
2.2	Схема многозадачности системы исполнения.....	77
2.2.1	Процесс «DI».....	77
2.2.1.1	Задача «CPU.EXTI».....	77
2.2.1.2	Задача «DI_IRQ_TASK».....	77
2.2.1.3	Задача «DI_SET_TASK».....	78
2.2.2	Процесс «DO».....	78
2.2.2.1	Задача «CPU.TIM».....	79
2.2.2.2	Задача «DO_SET_TASK».....	79
2.2.3	Процесс «AI».....	79
2.2.3.1	Задача «CPU.ADC».....	80
2.2.3.2	Задача «AI_TIM_TASK».....	81
2.2.3.3	Задача «AI_SET_TASK».....	81
2.2.4	Процесс «MODBUS».....	82
2.2.4.1	Задача «MODBUS_TASK».....	82
2.2.5	Процесс «DATA».....	83
2.2.5.1	Задача «DATA_TASK».....	83
2.2.6	Процесс «REGISTER CONTROLLER».....	85
2.2.7	Процесс «APP».....	85
2.2.7.1	Задача «APP_TASK».....	85
2.2.7.2	Задача «APP_TIM_TASK».....	86
2.3	Карта регистров данных.....	86
2.4	Выбор средств разработки.....	87
2.4.1	Встраиваемая операционная система реального времени.....	87
2.4.2	Стандартные библиотеки периферии микроконтроллера.....	90
2.4.3	Среда разработки.....	91
2.4.4	Графический конфигуратор микроконтроллера.....	92
2.4.5	Программатор.....	93
2.5	Разработка системы исполнения.....	94
2.5.1	Создание нового проекта.....	94

2.5.2	Подключение FreeRTOS.....	100
2.5.3	Подключение MatIEC.....	102
2.5.4	Код проекта.....	102
2.5.4.1	Конфигурация системы исполнения.....	102
2.5.4.2	Настройки системы тактирования микроконтроллера.....	103
2.5.4.3	Конфигурация FreeRTOS.....	103
2.5.4.4	Определение элементов ядра FreeRTOS.....	103
2.5.4.5	Работа с регистрами данных.....	105
2.5.4.6	Процесс «DI».....	105
2.5.4.7	Процесс «DO».....	106
2.5.4.8	Процесс «AI».....	106
2.5.4.9	Процесс «MODBUS».....	107
2.5.4.10	Процесс «DATA».....	107
2.5.4.11	Процесс «APP».....	107
2.5.4.12	Главная функция «main».....	108
2.5.5	Сценарий компоновщика кода.....	108
2.5.6	Файловая структура проекта.....	109
2.5.7	Сборка проекта.....	112
2.5.8	Загрузка исполняемого кода в микроконтроллер.....	113
2.6	Создание описания целевой платформы.....	114
	ЗАКЛЮЧЕНИЕ.....	115
	БИБЛИОГРАФИЧЕСКИЙ СПИСОК.....	116
	ПРИЛОЖЕНИЕ А КАРТА АДРЕСОВ РЕГИСТРОВ ДАННЫХ.....	119
	ПРИЛОЖЕНИЕ Б ФУНКЦИОНАЛЬНАЯ СХЕМА УСТРОЙСТВА.....	123
	ПРИЛОЖЕНИЕ В СХЕМА МНОГОЗАДАЧНОСТИ.....	124

## ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

АСУ ТП - автоматизированная система управления технологическим процессом.

В/В (I/O, Input/Output, Ввод/Вывод) - самостоятельное в логическом отношении устройство, обеспечивающее взаимодействие процессора с периферийными устройствами (датчиками, исполнительными механизмами, сетевыми устройствами и т. д.); синоним GPIO.

ЖК (LCD, Liquid Crystal Display, Жидкокристаллический индикатор) - экран на основе жидких кристаллов.

МЭК (IEC, International Electrotechnical Commission, Международная электротехническая комиссия) - международная некоммерческая организация по стандартизации в области электрических, электронных и смежных технологий.

НИОКР — научно-исследовательские и опытно-конструкторские работы.

ОЗУ (RAM, Random Access Memory, Оперативное Запоминающее Устройство, Оперативная память) - в большинстве своем энергозависимая память, в которой во время работы хранится исполняемый машинный код (программы) и обрабатываемые данные (константы, переменные).

ОС (OS, Operating System, Операционная Система) - совокупность ядра операционной системы и работающих поверх него программ и утилит, предоставляющих интерфейс для взаимодействия вычислительного устройства с периферией (включая пользователя).

ОСРВ (RTOS, Real-Time Operating System, Операционная Система Реального Времени) - тип специализированной операционной системы, основное назначение которой — предоставление необходимого и достаточного набора функций для проектирования, разработки и функционирования систем реального времени на конкретном аппаратном оборудовании; для подобных систем

					27.03.04.2023.203.23.05 ВКР	Лист
Изм.	Лист	№ докум.	Подпись	Дата		7

характерно: гарантированное (повышенное, "реактивное") время реакции на внешние события (прерывания от оборудования) и жесткая подсистема планирования процессов.

ПАЗ (Противоаварийная Защита) - аппаратно-программный комплекс, который используется в критических приложениях для перевода системы управления в безопасное состояние.

ПЗУ (Постоянное запоминающее Устройство) - энергонезависимая память, используется для хранения массива неизменяемых данных.

ПЛК (PLC, Programmable Logic Controller, Программируемый Логический Контроллер) - это промышленный компьютер (цифровая вычислительная система), который был усовершенствован и адаптирован для управления производственными процессами, такими как сборочные линии, станки, роботизированные устройства или любая деятельность, требующая высокой надежности, простоты программирования и диагностики технологических неисправностей.

ПО (Программное Обеспечение) - программа или множество программ, используемых для управления каким-либо вычислительным устройством (компьютер, микроконтроллер и т.п.).

ПР (Программируемое реле) - программируемый контроллер с ограниченными ресурсами, предназначенный для быстрой и легкой замены небольших релейных схем управления.

ШИМ (PWM, Pulse-Width Modulation) - процесс управления мощностью методом пульсирующего включения и выключения потребителя энергии.

ЦПУ (Central processing unit, CPU, Центральное Процессорное Устройство) - электронный блок либо интегральная схема, исполняющая

					27.03.04.2023.203.23.05 ВКР	Лист
Изм.	Лист	№ докум.	Подпись	Дата		8

машинные инструкции (код программ), главная часть аппаратного обеспечения компьютера или программируемого логического контроллера.

ADC (Analog-to-digital converter, Аналого-Цифровой Преобразователь) - устройство, преобразующее аналоговый сигнал в дискретный (цифровой) код.

AI (Analog Input, Аналоговый вход).

API (Application Programming Interface) - описание способов взаимодействия одной компьютерной программы с другими; обычно представляет собой набор функций.

ARM (Advanced RISC Machine, Усовершенствованная RISC-машина) - система команд и семейство описаний и готовых топологий 32- и 64-битных микропроцессорных/микроконтроллерных ядер, разрабатываемых компанией ARM Limited.

DI (Digital Input, Дискретный Вход).

DO (Digital Output, Дискретный Выход).

DMA (Direct Memory Access, Устройство Прямого Доступа к Памяти) - функция компьютерных систем, позволяющая определенным аппаратным подсистемам получать доступ к основной системной памяти независимо от центрального процессора.

EEPROM (Electrically Erasable Programmable Read-Only Memory) - один из видов энергонезависимой перезаписываемой памяти для постоянного хранения данных.

ETHERNET - Семейство технологий пакетной передачи данных между устройствами для компьютерных и промышленных сетей.

					27.03.04.2023.203.23.05 ВКР	Лист
Изм.	Лист	№ докум.	Подпись	Дата		9

FIFO (First In - First Out) - массив данных класса "очередь" с принципом "первым пришел - первым ушел".

FLASH (Flash Memory) - разновидность полупроводниковой технологии электрически перепрограммируемой памяти (EEPROM).

GCC (GNU Compiler Collection) - набор компиляторов для различных языков программирования и процессорных архитектур, разрабатываемый в рамках проекта GNU.

GNU (GNU Project) - ассоциация разработчиков свободного программного обеспечения.

GPIO (General-Purpose Input/Output) - интерфейс для связи между процессором и внешними периферийными устройствами; в качестве GPIO, обычно, выступают выводы процессора (микроконтроллера).

I2C (Inter-Integrated Circuit) - последовательная асимметричная шина для связи между интегральными схемами внутри электронных приборов; для связи использует 2 провода (SDA и SCL).

JTAG (Joint Test Action Group) - аппаратный интерфейс на базе стандарта IEEE-1149.1, предназначенный для подключения сложных цифровых микросхем к стандартной аппаратуре тестирования и отладки.

HART (Highway Addressable Remote Transducer) - набор коммуникационных стандартов для промышленных сетей, предназначенный для подключения промышленных датчиков и включающий проводной и беспроводной физические уровни, а также протокол обмена; проводной вариант позволяет передавать цифровые данные поверх аналогового сигнала 4-20 мА (токовая петля).

					27.03.04.2023.203.23.05 ВКР	Лист
Изм.	Лист	№ докум.	Подпись	Дата		10

HMI (Human-Machine Interface, Человеко-Машинный Интерфейс) - широкое понятие, охватывающее инженерные решения, обеспечивающие взаимодействие человека-оператора с управляемыми им машинами. Чаще всего HMI реализуется с использованием типовых средств: панели оператора, компьютеры и ПО.

MODBUS - открытый коммуникационный протокол, основанный на архитектуре ведущий — ведомый (master-slave); RTU - физический уровень RS-232, RS-485; TCP - физический уровень ETHERNET.

PROFIBUS (Process Field Bus, Шина Полевого Уровня) - открытая промышленная сеть, прототип которой был разработан компанией Siemens AG для своих промышленных контроллеров Simatic; описывает физический уровень и протокол передачи данных.

PROFINET (Process Field Network, Сеть Полевого Уровня) - открытый промышленный стандарт для автоматизации, использующий TCP/IP-стандарты и режим реального времени ETHERNET.

RISC (Reduced Instruction Set Computer, Вычислитель с Набором Упрощенных Команд) - архитектурный подход к проектированию процессоров, в которой быстродействие увеличивается за счет использования упрощенной системы и конвейера команд.

RTC (Real Time Clock, Часы Реального Времени) - электронная схема, предназначенная для учета хронометрических данных (текущее время, дата, день недели и др.) и включающая в себя микросхему учитывающее устройства и автономный источник питания.

RS-232 (Recommended Standard 232, EIA232) - стандарт физического уровня для асинхронного интерфейса (UART) с полнодуплексной линией связи типа "точка-точка"; в компьютерной технике известно как устройство последовательного порта (COM-порт).

RS-485 (Recommended Standard 485, EIA485) - стандарт физического уровня для асинхронного интерфейса (UART) с полудуплексной многоточечной дифференцированной линией связи типа "общая шина".

SCADA (Supervisory Control and Data Acquisition) - аппаратно-программный комплекс для диспетчерского управления и сбор данных.

SIL (Safety Integrity Level) - уровень полноты безопасности, определяющий требования к полноте функциональной безопасности для критичных по безопасности устройств.

SPI (Serial Peripheral Interface, Последовательный Периферийный Интерфейс) - последовательный синхронный стандарт передачи данных в режиме полного дуплекса типа "общая шина", предназначенный для обеспечения простого и недорогого высокоскоростного сопряжения микроконтроллеров и периферии; для связи используется 4 провода (MISO, MOSI, SCL, CS).

SRAM (Static RAM) - статическая ОЗУ с произвольным доступом, позволяющая поддерживать состояние без регенерации (которая необходима в динамической памяти).

UART (Universal Asynchronous Receiver-Transmitter, Универсальный Асинхронный Приемопередатчик) - узел вычислительных устройств, предназначенный для организации последовательной связи с другими цифровыми устройствами; для связи используется от 2 проводов (RX, TX).

USB (Universal Serial Bus) - универсальная последовательная шина - последовательный интерфейс для подключения периферийных устройств к вычислительной технике.

					27.03.04.2023.203.23.05 ВКР	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		



## ВВЕДЕНИЕ

Отечественный рынок автоматизации долгое время делился между такими крупными международными игроками как Siemens, Allen Bradley, Schneider Electric, Omron, Advantech, Yokogawa. Но, в текущих политических и экономических реалиях большинство этих производителей ушло с нашего рынка, введя запрет (санкции) на поставку своих технологий. В связи с этим, от отечественных предприятий стали часто поступать запросы по замене, модернизации или проектированию новых систем автоматизации с использованием отечественных решений. Появившийся спрос, а также государственная программа импортозамещения дают хорошие возможности развиваться отечественным разработчикам - крупным компаниям (таким как, например, Овен, Прософт, Сегнетикс, Тэкон, Агава, Адастра, Круг, Т-Софт), а также начинающим.

Цель данной работы: в рамках одного из заказов программы импортозамещения, в составе группы разработчиков электроники провести эксперимент по созданию бюджетной модели свободно программируемого контроллера (ПЛК) с использованием стандартных, открытых и доступных аппаратно-программных средств.

Условия эксперимента следующие:

- 1) аппаратная часть устройства должна быть построена на базе процессорной архитектуры (например, ядра RISC-V, ARM Cortex-M3, ARM Cortex-M4, MIPS, SPARC), разрабатываемой отечественными производителями (Миландр, Микрон, НИИЭТ, Ангстрем, Байкал Электроникс, МЦСТ); для быстрого старта допускается применение доступных в продаже процессоров иностранного производства, но с аналогичными архитектурами;
- 2) программная часть нижнего уровня (уровня ядра процессора) — встраиваемая система исполнения - должна реализовываться с применением стандартных, открытых и свободно доступных средств разработки;
- 3) программная часть верхнего уровня (уровня инженера по автоматизации) — встраиваемая управляющая программа - должна реализовываться с применением

					27.03.04.2023.203.23.05 ВКР	Лист
Изм.	Лист	№ докум.	Подпись	Дата		13

открытых, свободно доступных, аппаратнонезависимых средств разработки, соответствующих стандарту МЭК-61131-3.

На начальном этапе эксперимента, схемотехниками группы был разработан и изготовлен отладочный экземпляр аппаратной платформы под названием «ПЛК411», основой которой является микроконтроллер с ядром архитектуры ARM Cortex-M4 (для быстрого старта использовался имеющийся в наличии микроконтроллер STM32F411). Системотехники и программисты выполнив анализ существующих средств разработки и, в соответствии с заявленными требованиями, выбрали среду программирования Beremiz. В итоге, осталось реализовать программную часть нижнего уровня, для чего совместными усилиями специалистов были сформулированы требования, оформленные в виде задания на разработку встраиваемой системы исполнения.

Таким образом, задачи данной работы следующие:

- 1) изучить основные классы современных ПЛК;
- 2) изучить типы программного обеспечения, встраиваемого в ПЛК;
- 3) изучить базовую логику работы стандартного ПЛК;
- 4) изучить основы стандарта МЭК-61131-3;
- 5) изучить логику создания управляющей программы в среде Beremiz, способы ее загрузки в ПЛК и возможные механизмы ее взаимодействия с системой исполнения ПЛК (как управляющая программа будет исполняться в ПЛК и работать с его периферией);
- 6) изучить характеристики микроконтроллера STM32F411;
- 7) выполнить подробный анализ требований к системе исполнения;
- 8) выбрать необходимые средства разработки;
- 9) разработать функциональную схему контроллера;
- 10) разработать схему многозадачности системы исполнения;
- 11) разработать карту адресов регистров данных;
- 12) запрограммировать систему исполнения;
- 13) проверить работоспособность разработанной системы исполнения.

					27.03.04.2023.203.23.05 ВКР	Лист
Изм.	Лист	№ докум.	Подпись	Дата		14

# 1 ОБЗОРНЫЙ РАЗДЕЛ

## 1.1 Введение в стандартный ПЛК

### 1.1.1 Назначение и структура

Программируемый логический контроллер (ПЛК) – специализированное микропроцессорное устройство со встроенным аппаратным и программным обеспечением, которое используется для выполнения функций управления технологическим оборудованием. [1]

Особенности ПЛК, отличающие его от прочих электронных приборов:

- является полностью самостоятельным устройством;
- ориентирован на работу с электронным оборудованием через развитый ввод сигналов датчиков и вывод сигналов на исполнительные механизмы;
- обеспечивает реальное время — когда система реагирует на внешние воздействия за минимально короткий период времени (без задержек).

ПЛК, главным образом, состоит из следующих компонентов (рисунок 1.1):

- блок питания,
- центральное процессорное устройство (ЦПУ, микроконтроллер),
- область памяти (ОЗУ и память программ),
- каналы ввода/вывода (В/В),
- сетевые интерфейсы,
- человеко-машинный интерфейс (световая индикация, кнопки, ЖК-экран).

Каналы ввода обеспечивают связь ПЛК с внешними датчиками. Различают аналоговые и дискретные входы, предназначенные для работы с аналоговыми и дискретными сигналами соответственно.

ЦПУ – «мозг» ПЛК, осуществляющий его логику работы. Это процессор, обрабатывающий команды программы и управляющий всеми внутренними элементами: входами, выходами, сетевыми интерфейсами, индикацией.

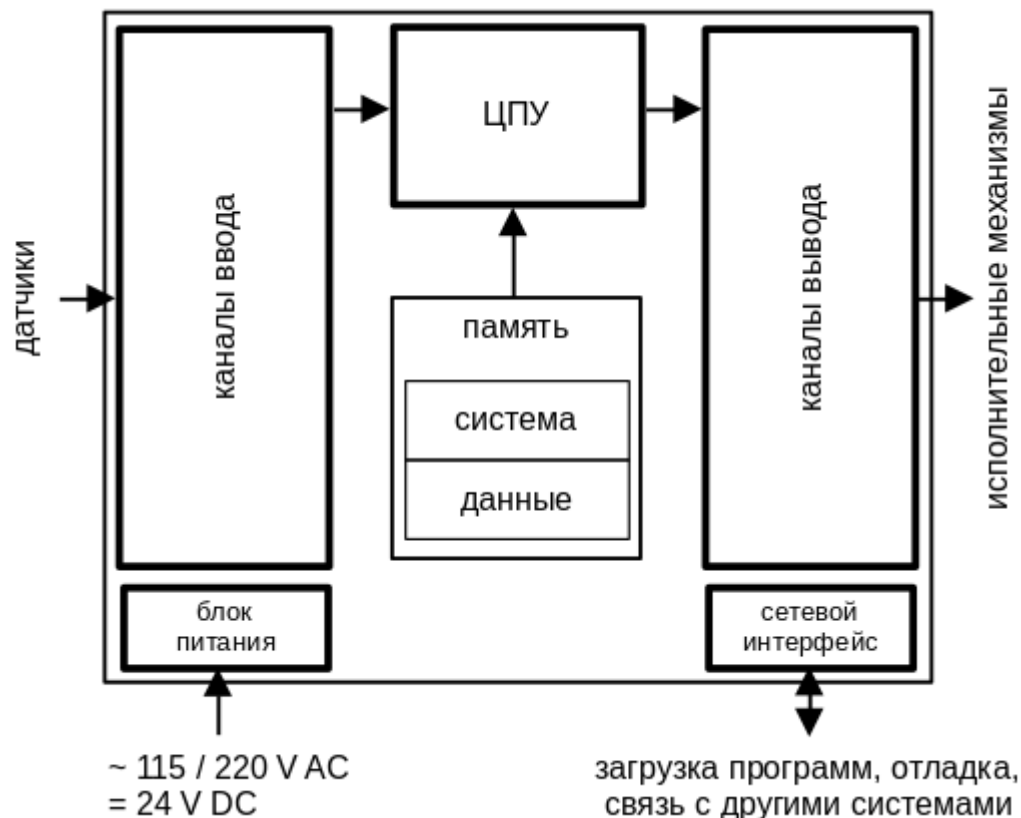


Рисунок 1.1 - Структурная схема ПЛК

ПЛК обладает некоторым объемом памяти, которая в различных моделях может иметь различную организацию. Как правило, выделяют следующие разделы памяти:

- оперативная память (ОЗУ) — энергозависимая память, куда загружается программа и где хранятся данные во время работы контроллера,
- память программ (FLASH, EEPROM, MMC) — энергонезависимая память, где хранится исполняемая программа.

Каналы вывода обеспечивают связь ПЛК с внешними исполнительными механизмами. Как и входы, различают аналоговые и дискретные выходы, которые предназначены для работы с соответствующими сигналами.

Сетевой интерфейс (USB, RS-232, RS-485, ETHERNET) обеспечивает подключение ПЛК к какой-либо внешней периферии:

- к ПК для загрузки или отладки программы,
- к какой-либо системе для обмена данными.

Блок питания предназначен для обеспечения работы ПЛК, а также может быть использован для питания внешних входных и выходных цепей. Блок питания может быть как внешнего, так и внутреннего исполнения (является частью аппаратной конструкции ПЛК).

Как таковые, первые ПЛК появились в 1969 году и применялись на автомобильном заводе GeneralMotors для замены релейных схем управления [2]. Современные ПЛК, использующие инновационные технологии, далеко ушли от первых упрощенных реализаций промышленного контроллера, но заложенные в систему управления универсальные принципы были стандартизированы и успешно развиваются уже на базе новейших технологий.

### 1.1.2 Классификация контроллеров

В зависимости от конструктивного исполнения и функционала, можно выделить следующие виды ПЛК (рисунок 1.2):

- интеллектуальные (программируемые) реле,
- моноблочные контроллеры,
- модульные контроллеры,
- распределенные системы.

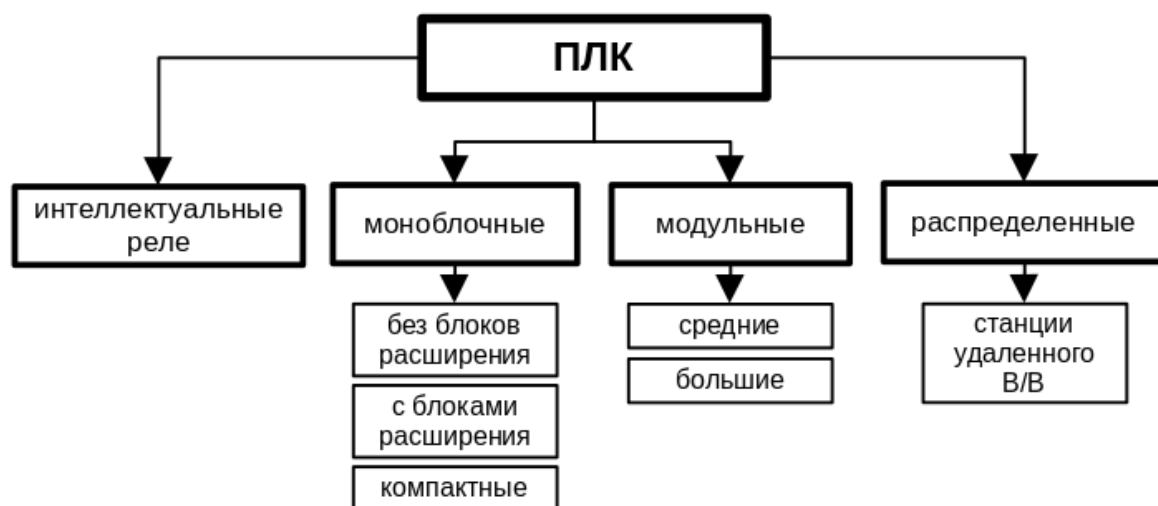


Рисунок 1.2 - Виды ПЛК

### 1.1.2.1 Интеллектуальные реле

Интеллектуальное реле (ПР, программируемое реле) — устройство (младший подкласс ПЛК), предназначенное для быстрой и легкой замены небольших релейных схем управления; с помощью одного ПР можно заменить схему управления, состоящую из группы реле (рисунок 1.3 [3][4]).

Основные характеристики ПР:

- маломощный микроконтроллер;
- небольшой объем памяти ОЗУ и памяти программ;
- ограниченное количество каналов В/В (до 4-х дискретных, до 2-х аналоговых);
- сетевой интерфейс программирования (для загрузки и отладки управляющей программы; например, USB);
- человекомашинный интерфейс (светодиодная индикация, кнопки, небольшой ЖК-экран);
- ограниченный набор программируемых функций (дискретная / релейная логика).



(а)



(б)

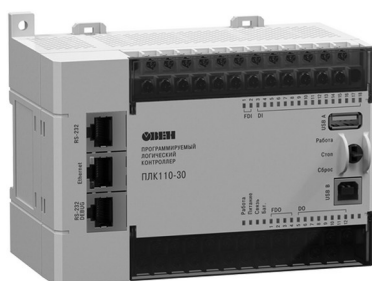
Рисунок 1.3 - Овен ПР110 (а) и ПР200 (б)

### 1.1.2.2 Моноблочные контроллеры

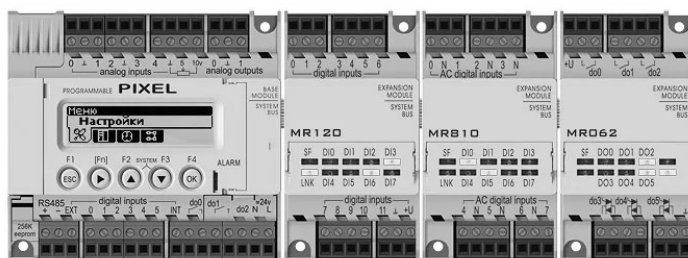
Моноблочный ПЛК — устройство (средний подкласс ПЛК), по сути, являющееся программируемым реле с расширенным аппаратно-программным функционалом, где все блоки находятся в одном корпусе (рисунок 1.4 [5][6]).

### Основные характеристики моноблочного ПЛК:

- микроконтроллер средней или большой мощности;
- увеличенная память (ОЗУ и память программ);
- простой человеко-машинный интерфейс (светодиодная индикация, кнопки);
- увеличенное количество каналов В/В (до 24 дискретных, до 8 аналоговых);
- сетевой интерфейс программирования (например, RS-485, USB, ETHERNET - для загрузки и отладки управляющей программы);
- сетевой интерфейс передачи данных с поддержкой стандартных протоколов (например, до 2-х RS-485, ETHERNET, протоколы — MODBUS и т. п.);
- расширенный и разнообразный набор программируемых функций;
- возможность расширения функционала путем подключения внешних модулей (модулей / блоков расширения).



(a)



(б)

Рисунок 1.4 - Моноблочный Овен ПЛК110 (а)  
и ПЛК Segnetics Pixel с подключаемыми модулями расширения (б)

### Недостатки моноблочной конструкции:

- избыточность (например, могут быть задействованы не все каналы);
- низкая ремонтнопригодность и взаимозаменяемость (например, для ремонта вышедшего из строя канала В/В ПЛК необходимо выводить из работы ПЛК в целом, а в некоторых случаях выход из строя одного канала может вывести из строя весь ПЛК, что влечет за собой остановку управляемого технологического процесса).

#### 1.1.2.3 Модульные контроллеры

Модульный ПЛК по основным характеристикам аналогичен моноблочным решениям, но отличается конструктивным исполнением. Данный класс ПЛК

состоит из следующих основных компонентов: базовый модуль и шина (крейт) для подключения функциональных модулей (модулей расширения).

Базовый модуль, обычно, включает в себя:

- процессорное устройство;
- память (ОЗУ и программ, опционально — EEPROM, карта памяти);
- человекомашинный интерфейс;
- сетевой интерфейс программирования;
- блок питания (может отсутствовать);
- интерфейс связи (может отсутствовать).

Конструкция шины (крейта) зависит от производителя и модели ПЛК, но, обычно — это плата с разъемами (слотами) для крепления функциональных модулей (рисунки 1.5 (а, б) [7][8]). Как правило, через шину функциональные модули получают питание и связь с базовым модулем. Шина может компоноваться модулями не полностью (могут быть свободные слоты — в некоторых моделях ПЛК закрываются специальными заглушками). Максимальное количество подключаемых на шину модулей зависит от модели ПЛК (например, ПЛК SIEMENS серии S7-400 позволяет устанавливать на шину до 48 функциональных модулей).

Выделяют следующие функциональные модули:

- каналы дискретного ввода (DI),
- каналы дискретного вывода (DO),
- каналы аналогового ввода (AI),
- каналы аналогового вывода (AO),
- каналы унифицированные (например, аналоговые 4-20 мА+HART, 0-10 В),
- резервный / дублирующий базовый,
- резервный / дублирующий блок питания,
- интерфейсы связи (RS-232, RS-485, ETHERNET, оптика и т. д.),
- дополнительная память (EEPROM),
- прочие специального назначения (например, весоизмерительный).



Производитель функциональных модулей, обычно, тот же, кто и производит сам ПЛК. Каждый функциональный модуль, как правило, имеют всю необходимую (схемотехническую) защиту, чтобы при выходе из строя не повлиять на работу базы и остальных модулей шины. Некоторые модели ПЛК поддерживают «горячую замену» модулей — замену модуля без выключения питания ПЛК, а также функции резервирования / дублирования (обычно используются в безотказных системах управления — например, системы противоаварийной защиты — ПАЗ, рисунок 1.5 (в) [9]).

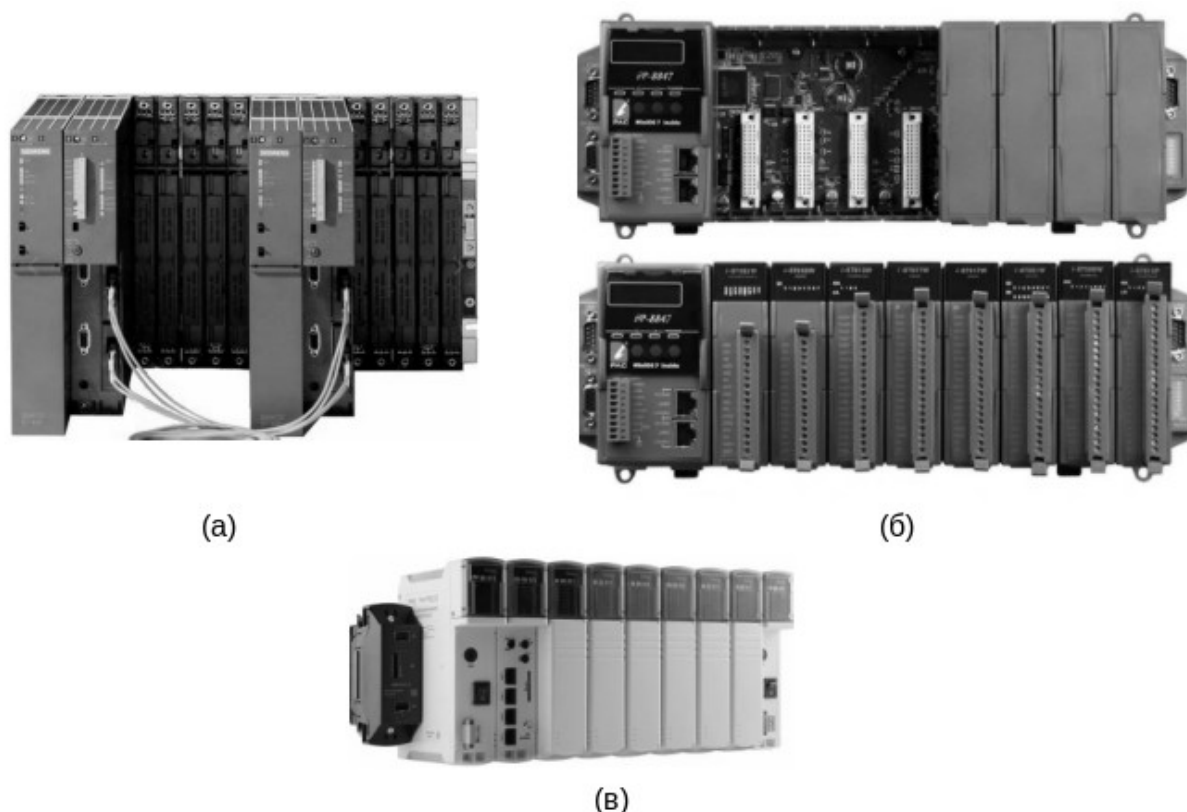


Рисунок 1.5 - ПЛК SIEMENS S7-410H с резервированием базы и питания (а), ПЛК ICP DAS iP-8847 (б) и ПЛК ПРОСОФТ REGUL R500S для систем ПАЗ (в)

Достоинства модульной конструкции ПЛК:

- низкая избыточность функционала (компоновка под конкретную задачу);
- высокая ремонтнопригодность и взаимозаменяемость

(например, для ремонта вышедшего из строя канала В/В достаточно заменить соответствующий модуль, а при поддержке «горячей замены» это происходит без отключения питания и, соответственно, без остановки управляемого технологического процесса).

#### 1.1.2.4 Распределенные системы

Распределенные системы управления строятся на устройствах подкласса ПЛК, куда входят: центральные ПЛК и устройства (станции) удаленного В/В.

Центральный ПЛК (обычно модульного типа) включает: базовый модуль, модуль блока питания и модуль сетевых интерфейсов (с резервированием или без). ПЛК, как правило, располагается на удалении от объекта управления (например, в шкафу в электропомещении, помещении серверной или операторской).

Весь функционал В/В осуществляется отдельными устройствами - станциями В/В, которые удалены от центрального ПЛК и располагаются, обычно, в непосредственной близости от объекта управления.

Связь центрального ПЛК и станции В/В выполняется по одному из цифровых интерфейсов (RS-485, ETHERNET, оптика).

Станции В/В выполняют следующие функции:

- сбор данных с датчиков объекта управления (каналы ввода),
- преобразование сигналов из одной формы в другую (АЦП, ЦАП),
- передача/прием данных в цифровом виде по интерфейсам связи,
- выдача сигналов на объекты управления (каналы вывода),
- диагностика каналов В/В и связи.

Центральные ПЛК выполняют следующие функции:

- сбор данных в цифровом виде с станций В/В,
- исполнение алгоритмов управления,
- передачу управляющих команд в цифровом виде на станции В/В,
- предоставление данных в системы диспетчеризации (SCADA).

Обычно, производители выпускают станции В/В, совместимые на уровне модулей с линейкой производимых ПЛК (например, станции SIEMENS ET-200, для которых могут применяться функциональные модули от ПЛК SIEMENS S7-

					27.03.04.2023.203.23.05 ВКР	Лист
Изм.	Лист	№ докум.	Подпись	Дата		22

300/400) или предоставляющие стандартный интерфейс и протокол связи (RS-485 / ModBus RTU / ProfiBus, ETHERNET / ModBus TCP / ProfiNet). Некоторые производители предлагают целый комплекс аппаратно-программных средств, реализующий распределенные системы управления различного масштаба (например, SIEMENS PCS7, рисунок 1.6 [10]).

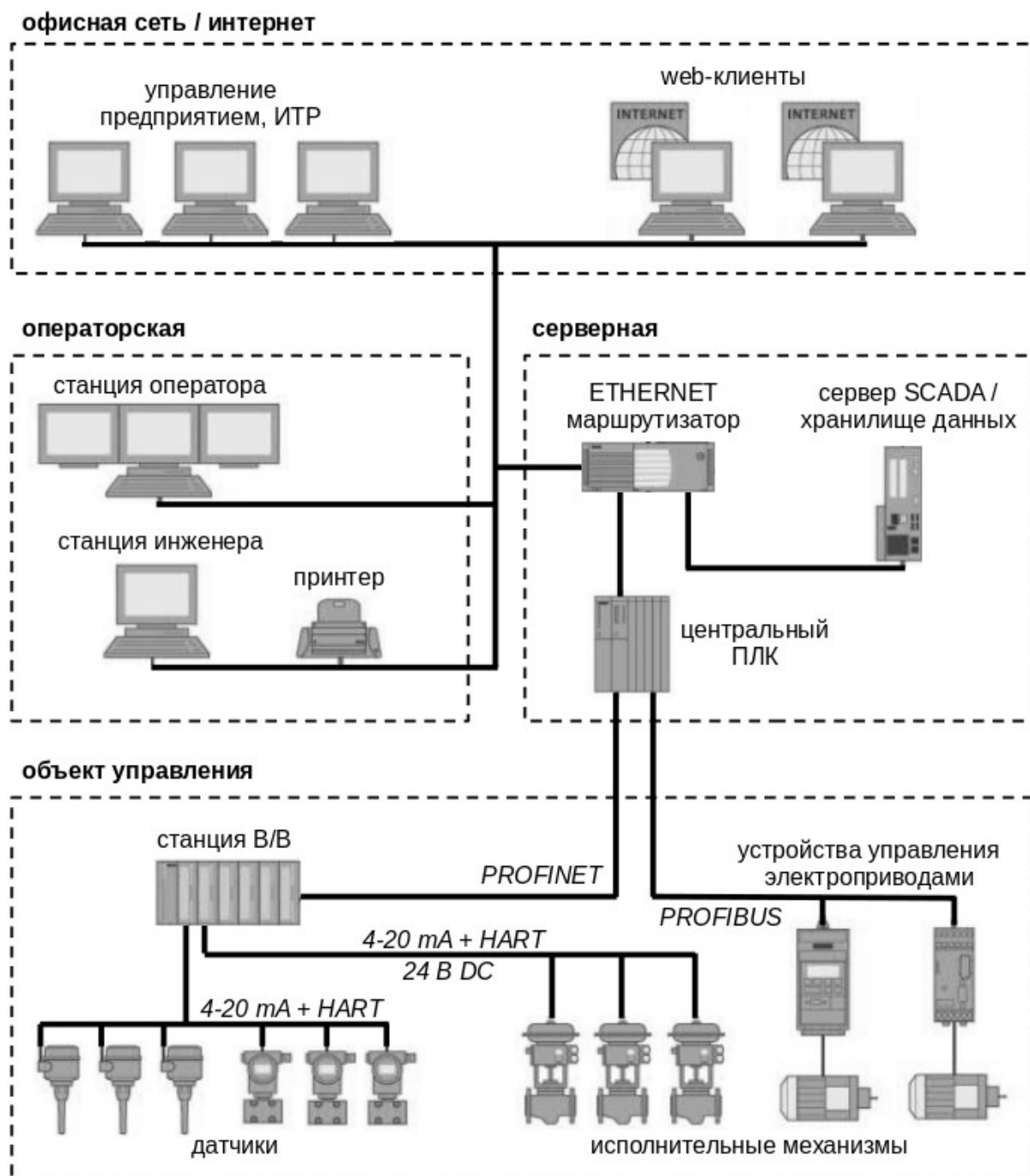


Рисунок 1.6 - Пример распределенной системы управления

### 1.1.3 Встраиваемое программное обеспечение

Поскольку ПЛК строится на основе цифровой техники, естественным образом предполагается наличие программного обеспечения (ПО) и языковых средств программирования, с помощью которых и реализуется, собственно, логика работы самого ПЛК и логика системы управления технологическим процессом. В силу специфики микропроцессорной техники данное ПО хранится в энергонезависимой памяти контроллера и поэтому называется встраиваемым ПО (в некоторых источниках — встроенным).

Обычно встраиваемое ПО специализировано для конкретного устройства и имеет, соответствующие ему, ограничения по ресурсам: размер доступной памяти и быстродействие.

Функционально встраиваемое ПО разделяют на:

- системное (или система исполнения / целевая система),
- пользовательское (или управляющая программа).

#### 1.1.3.1 Система исполнения

Система исполнения — системное ПО нижнего уровня (уровня процессора и его периферии), которое реализует базовую логику работы ПЛК.

Система исполнения разрабатывается и поставляется производителем ПЛК. В силу специфики, реализация системы исполнения выполняется на языках программирования низкого и среднего уровня (обычно, язык Си).

Функции системы исполнения:

- аппаратные:
  - работа с каналами В/В и сетевыми интерфейсами,
  - работа с памятью;
- системные:
  - событийность (обработка прерываний / событий),
  - служба времени (аппаратные и программные таймеры),

- логический параллелизм (многозадачность);
- взаимодействие с управляющей программой (загрузка/обновление, отладка, исполнение).

Система исполнения может быть реализована следующими методами: [12]

- Bare Metal («Без операционной системы»):
  - основа — бесконечный главный цикл («main loop»),
  - многозадачность реализуется обработчиками аппаратных прерываний,
  - жесткое, гарантированное время отклика на прерывания.
- RTOS (ОСРВ, операционная система реального времени):
  - основа — ОСРВ для микроконтроллеров, предоставляющая набор готовых функций операционной системы,
  - многозадачность реализуется средствами ОСРВ и обработчиками аппаратных прерываний,
  - жесткое, гарантированное время отклика на прерывания.
- OS (ОС, операционная система):
  - основа — ОС общего назначения (например, Linux, Android, Windows),
  - многозадачность реализуется средствами ОС,
  - мягкое, с задержками время отклика на прерывания (с помощью специальных программных средств возможна реализация жесткого времени отклика — например, специальная сборка ядра Linux или использование «микроядра RTOS» совместно с «мягким» ядром Linux – Linux+Xenomai).

### 1.1.3.2 Управляющая программа

Управляющая программа — пользовательское ПО верхнего уровня (уровня инженера АСУ ТП), которое реализует непосредственно алгоритм системы управления технологическим процессом: оперируя различными средствами специального языка программирования и специальными функциями (системные вызовы системы исполнения), взаимодействующими с периферией ПЛК.

Предполагается, что разработкой управляющей программы занимается специалист, непосредственно проектирующий или эксплуатирующий систему

					27.03.04.2023.203.23.05 ВКР	Лист
Изм.	Лист	№ докум.	Подпись	Дата		25

управления технологическим процессом — это пользователь / потребитель ПЛК (инженер АСУ ТП, инженер-системотехник, инженер-технолог). В силу данной специфики, использование здесь языков программирования низкого и среднего уровня не подходит. Поэтому, для написания управляющей программы применяются простые в изучении и понимании специальные языки программирования верхнего уровня (языки промышленной автоматизации) — языки стандарта МЭК-61131-3 [1]. Данные языки структурированы таким образом, чтобы программирование велось в естественных терминах технологического процесса.

#### 1.1.4 Базовая логика работы

Прародителями ПЛК были релейные схемы автоматики. Это "родство" проявляется в виде жесткой цикличности работы его программ.

Разберем базовый алгоритм работы ПЛК (рисунок 1.7): в процессе работы ПЛК непрерывно опрашивает текущее состояние каналов ввода  $X_1...X_n$  и в соответствии с требованиями технологического процесса изменяет состояние каналов вывода  $Y_1...Y_n$ . [1]

Базовый алгоритм работы ПЛК можно разделить на несколько шагов: [1]

1) Инициализация системы. Необходимо помнить, что в случае сбоя по питанию или при выключении ПЛК система обязана вернуться в исходное состояние. Не следует недооценивать важности этой части программного кода, так как в противном случае это может привести к сбоям и поломкам технологического оборудования.

2) Чтение текущего состояния каналов ввода. Система исполнения ПЛК считывает текущее состояние каналов ввода. Состояние любого из считанных каналов сохраняется в памяти (регистры данных) и может в дальнейшем использоваться при обработке третьего шага программы.

3) Выполнение управляющей программы. Система исполнения ПЛК запускает на выполнение управляющую программу, передав ей обновленные состояния каналов ввода ПЛК и иные необходимые данные из регистров (например, данные, полученные через сетевой интерфейс и результаты работы алгоритма на

предыдущем цикле). Управляющая программа построчно выполняет заложенный в ней алгоритм управления технологическим процессом: обрабатывает входные данные и формирует управляющие воздействия. Результат обработки и управляющие воздействия сохраняются в памяти (регистры данных).

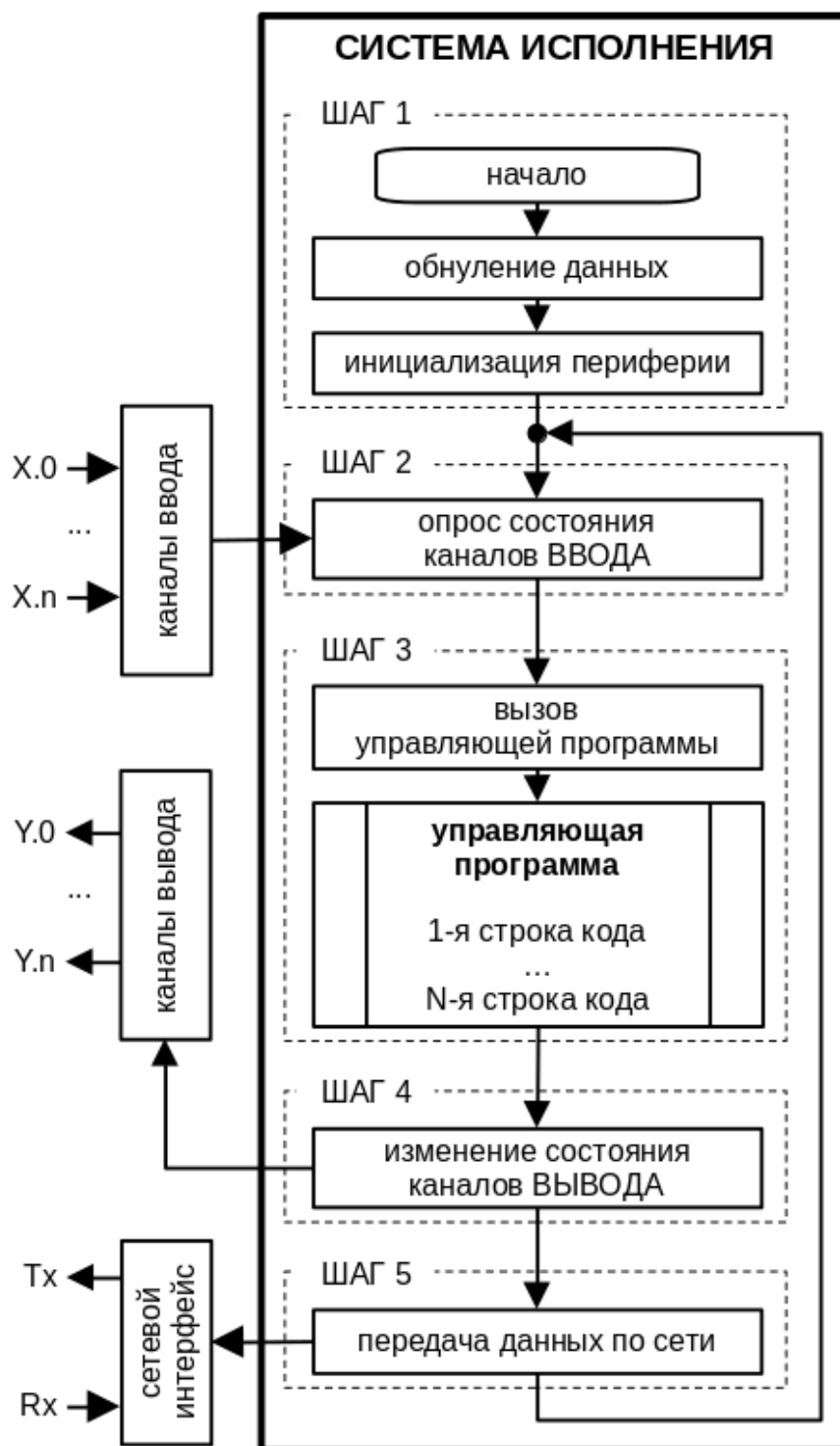


Рисунок 1.7 - Базовый алгоритм работы ПЛК

4) Изменение текущего состояния каналов вывода. Система исполнения ПЛК считывает значения управляющих воздействий из памяти (регистры данных) и записывает их в каналы вывода.

5) Передача данных через сетевой интерфейс (например, отправка данных в систему диспетчеризации).

На этом завершается один рабочий цикл и далее следует возврат к шагу 2 для выполнения следующего цикла. И так происходит, пока не будет выполнена перезагрузка ПЛК.

## 1.2 Операционная система реального времени

При написании программы без операционной системы (программа класса «Bare Metal»), так или иначе, все равно приходится создавать некоторое подобие собственной системы: планировщик, обработчики прерываний и событий. Создавать «собственную» операционную систему довольно сложное и, порой, затратное занятие. Поэтому в этой части, целесообразно выбрать какое-то готовое решение.

Готовая операционная система (ОС) предоставляет необходимый и достаточный набор средств: хорошо отлаженный и готовый к работе функционал, документацию, а в некоторых случаях и все необходимые сертификаты и техническую поддержку. Тем самым, программисту предоставляется возможность сконцентрироваться на решении конкретных проблемно-ориентированных задач, не отвлекаясь на системные задачи, которые берет на себя ОС. Использование ОС улучшает читабельность и наглядность программного кода. Каждую задачу в простом случае можно программировать отдельно, всю работу разбить между несколькими членами команды. Не нужно заботиться о переключении между задачами, это делает готовый планировщик.

Необходимо помнить, что ОС требует для работы ресурсы:

- дополнительный расход памяти программ для хранения ядра,
- дополнительный расход памяти данных для хранения стека каждой задачи и объектов ядра,



- дополнительные затраты времени процессора на переключение между задачами (время, необходимое для сохранения состояния задачи и ее последующего запуска с момента остановки).

Поэтому, если аппаратная платформа, для которой реализуется проект, имеет ограниченные ресурсы, то стоит предварительно осмыслить необходимость использования ОС.

В настоящее время на мировом рынке операционных систем присутствует достаточное для выбора количество решений, которые можно разделить на следующие классы:

- по типу реального времени:
  - общего назначения без реального времени,
  - общего назначения с микроядром реального времени,
  - реального времени;
- по аппаратной совместимости:
  - аппаратнозависимые (Intel-совместимые: для ПК и серверов),
  - аппаратнонезависимые (под различные процессорные архитектуры);
- по доступности:
  - коммерческие (платная лицензия),
  - свободно распространяемые (открытая, бесплатная лицензия).

Есть два основных вида операционных систем: общего назначения (general) и реального времени (real-time). Системы общего назначения — это обычные операционные системы в привычном нам смысле: Windows, Linux и Mac OS. Они решают много задач сразу и реакция на какие-либо события в них обрабатывается с задержкой (отложенный вызов обработчика событий), что порой выглядит как «зависание» или «притормаживание». Но есть устройства и задачи, где необходима моментальная реакция без права на ошибку или задержку. В них используют операционные системы реального времени (ОСРВ).

ОСРВ предоставляет:

- гарантированное время исполнения задач и обработчиков прерываний,

					27.03.04.2023.203.23.05 ВКР	Лист
Изм.	Лист	№ докум.	Подпись	Дата		29

- предсказуемое поведение при всех сценариях нагрузки,
- многозадачность и параллельное исполнение задач,
- готовый планировщик,
- готовые средства межпроцессного взаимодействия,
- готовые средства работы с общими ресурсами,
- программные таймеры.

### 1.2.1 Многозадачность

Основой ОСРВ является ядро (Kernel), позволяющее выполнять множество программ одновременно. Каждая программа представляет собой задачу (Task). Если ОСРВ позволяет одновременно выполнять множество задач, она является многозадачной (Multitasking). [15]

Задача — это подпрограмма с бесконечным циклом, работающая независимо от остальных подпрограмм. Выход из бесконечного цикла завершает работу задачи. Задача может уничтожать саму себя или любую другую задачу в системе. Нет возможности восстановить удаленную задачу, единственный выход — создать и запустить ее заново. Каждая задача имеет приоритет. Чем выше приоритет, тем быстрее эта задача будет выполнена. Если в системе есть задачи с одинаковым приоритетом, то в первую очередь будет исполнена та, которая дольше всех находится в состоянии ожидания. [15]

Задача может находиться в следующих состояниях:

- не выполняется:
  - приостановлена (suspended),
  - готова (ready),
  - блокирована (blocked);
- выполняется (running).

Задача находится в блокированном состоянии, если она ожидает наступление временного или внешнего события (event). Например, задача может заблокировать сама себя на определенное время (выполнить задержку) или быть

					27.03.04.2023.203.23.05 ВКР	Лист
Изм.	Лист	№ докум.	Подпись	Дата		30

заблокирована ядром при попытке доступа к общему ресурсу, который не был освобожден ранее другой задачей (например, к последовательному порту). [15]

При переходе от одной задачи к другой ядро сохраняет состояние приостанавливаемой задачи (сохраняются локальные переменные этой задачи). Для ОСРВ из ОЗУ (области «кучи») выделяется память — стек. Из стека каждой задаче выделяется необходимый объем для хранения ее состояния (контента — стек задачи, блока управления задачей — Task Control Block - TCB). Пример распределения памяти ОСРВ приведен на рисунке 1.8 [16]:

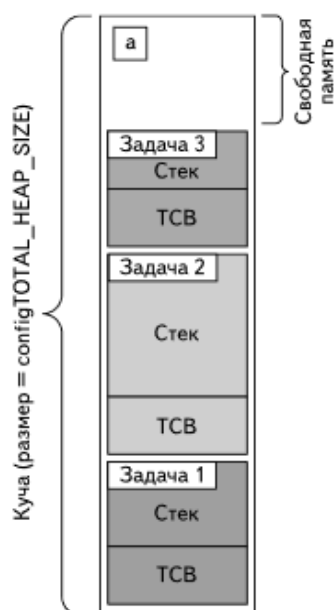


Рисунок 1.8 - Память ОСРВ

## 1.2.2 Планировщик

Большинство процессоров могут выполнять только одну задачу в один момент времени. Однако при помощи быстрого переключения между задачами достигается эффект параллельного выполнения всех задач: каждая задача выполняется определенное время, после чего процессор «переключается» на следующую задачу. Распределением процессорного времени между задачами выполняет планировщик (Scheduler), который является частью ядра ОСРВ.

На рисунке 1.9 показано истинно параллельное выполнение трех задач, на рисунке 1.10 - реальная реализация многозадачности [17].

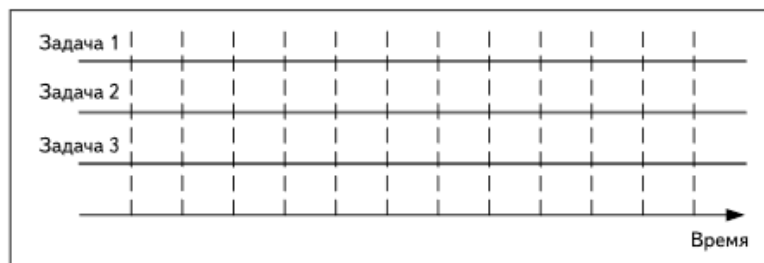


Рисунок 1.9 - Пример истинно параллельного выполнения трех задач

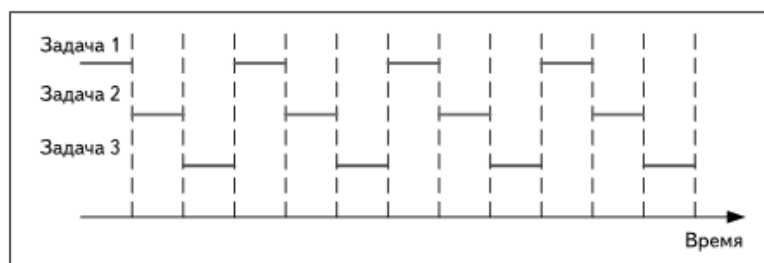


Рисунок 1.10 - Пример реального параллельного выполнения трех задач

Каждая задача выполняется определенный квант времени (tick) – это жестко фиксированный отрезок времени, когда планировщик не вмешивается в выполнение задачи. По истечении кванта времени планировщик может приостановить выполнение текущей задачи и возобновить следующую, готовую к исполнению. Для отсчета квантов в микроконтроллере обычно используется прерывание от таймера (обычно системного). Уменьшая продолжительности кванта, можно добиться более быстрой реакции системы на внешние события, однако это приведет к увеличению частоты вызова планировщика, что скажется на производительности вычислительной системы в целом. [17]

По принципу работы выделяют следующие виды многозадачности: вытесняющая или гибридная.

Вытесняющая многозадачность:

- каждой задаче назначается приоритет,
- каждая задача может находиться в одном из нескольких состояний,
- в один момент времени только одна задача может находиться в состоянии выполнения,
- планировщик переводит в состояние выполнения готовую задачу с наивысшим приоритетом,

- задачи могут ожидать наступления события, находясь в состоянии блокировки,
- планировщик получает управление каждый квант времени.

Гибридная (или кооперативная) многозадачность — когда планировщик не получает управление каждый квант времени, а вместо этого задача сама может вызвать планировщик — передать ему управление, не дожидаясь завершения выделенного ей кванта времени (рисунок 1.11). [16]

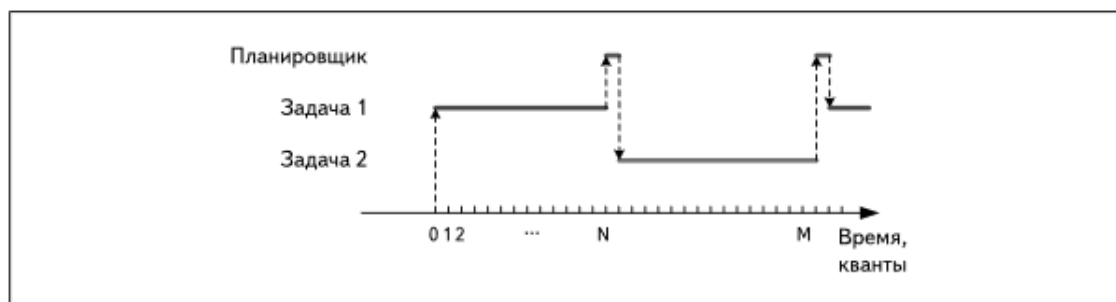


Рисунок 1.11 - Пример кооперативной многозадачности в ОСРВ

Принудительная передача управления планировщику из задачи осуществляется с помощью специальных API-функций. Результатом вызова такой API-функции может быть переключение на другую задачу или отсутствие переключения, если нет задач готовых к выполнению.

### 1.2.3 События и прерывания

Обычно встраиваемые системы обслуживают события, которые приходят от множества источников, причем каждое событие имеет свое требование по времени реакции системы и расходам времени на его обработку.

Прерывание (interrupt) — это событие (сигнал), заставляющее микроконтроллер изменить текущий порядок исполнения команд. При этом выполнение текущей последовательности команд приостанавливается, и управление передается обработчику прерывания — подпрограмме. Обработчик прерывания реагирует на событие и обслуживает его, после чего возвращает управление в прерванный код. Прерывания инициируются периферией

микроконтроллера, например прерывание от таймера/счетчика или изменение логического уровня на одном из входов микроконтроллера.

При проектировании встраиваемой микроконтроллерной системы на основе ОСРВ необходимо учесть, насколько долго продолжается процесс обработки прерывания. В самом простом случае, когда при обработке прерывания повторные прерывания запрещены, временные задержки в обработчике прерываний могут существенно ухудшить время реакции системы на события. Тогда для выполнения продолжительных действий по обработке прерывания вводится так называемый «отложенный» режим их выполнения. В процессе реакции на прерывание обработчик прерывания выполняет только первичные действия, например считывает данные и передает в очередь. Затем львиную долю обработки берет на себя задача-обработчик прерывания. Такая организация обработки прерываний называется отложенной обработкой. При этом обработчик прерывания выполняет только самые «экстренные» действия, а основная обработка «откладывается», пока ее не выполнит задача-обработчик. [18]

Если прерывание происходит при возникновении особенно критичного к времени реакции внешнего события, то имеет смысл назначить задаче-обработчику достаточно высокий приоритет, чтобы при возникновении прерывания она вытесняла другие задачи в системе.

#### **1.2.4 Межпроцессное взаимодействие и общие ресурсы**

Для безопасного взаимодействия между задачами, синхронизации прерываний и задач, а также при работе с одним и тем же ресурсом (например, одна область памяти) — в ОСРВ используются специальные механизмы: очереди, семафоры, мьютексы, критические секции.

##### **1.2.4.1 Очередь**

Очередь (queue) — это блок памяти фиксированного размера, представляющий собой массив элементов. Элемент очереди (item) — любой блок памяти от одного до нескольких байт (переменные, массивы, структуры,

указатели). Очередь может хранить в себе конечное число элементов фиксированного размера. Максимальное количество элементов в очереди — размер очереди. Размер элемента и очереди задаются при создании очереди.

Очередь функционирует по принципу «первым вошел — первым вышел» (FIFO), то есть элемент, который был помещен в очередь раньше, будет прочитан из очереди раньше. Элементы записываются в конец очереди, а считываются с начала [19]. При записи элемента в очередь, размер ее свободного места уменьшается, а при чтении — увеличивается (извлекаемый элемент из очереди удаляется).

Очередь является самостоятельным объектом и не принадлежит конкретной задаче. Как правило, несколько задач могут одновременно писать в очередь, а какая-то одна читать из нее. Когда задача пытается прочитать данные из очереди, которая не содержит ни одного элемента, то задача переходит в блокированное состояние. Такая задача вернется в состояние готовности к выполнению, если другая задача поместит в эту очередь данные — то есть появится какое-то событие (появление данных в очереди). Выход из блокированного состояния возможен по таймауту (если таймаут задан 0, то задача не блокируется). Как и при чтении из очереди, задача может находиться в блокированном состоянии, ожидая возможности записи в очередь. Это происходит, когда очередь полностью заполнена и в ней нет свободных мест, до тех пор, пока какая-либо задача не прочитает данные из очереди и, тем самым, не освободит в ней место.

#### 1.2.4.2 Семафор

Семафор (semaphore) – примитив синхронизации работы задач в ОСРВ, в основе которого лежит счетчик, над которым можно производить две атомарные операции: увеличение (инкремент) и уменьшение (декремент). При попытке уменьшения семафора, значение которого равно нулю, задача, запросившая данное действие, должна блокироваться до тех пор, пока не станет возможным уменьшение значения семафора до неотрицательного значения, то есть пока другой процесс не увеличит значение семафора. Под блокированием задачи

					27.03.04.2023.203.23.05 ВКР	Лист
Изм.	Лист	№ докум.	Подпись	Дата		35

понимается изменение состояния процесса или потока планировщиком задач на такое, при котором задача приостановит своё исполнение.

Семафоры позволяют переводить задачу из заблокированного состояния в состояние готовности каждый раз, когда происходит прерывание. Это дает возможность перенести большую часть кода, отвечающего за обработку прерывания, из обработчика прерывания в тело задачи. Внутри обработчика прерывания остается лишь небольшой, быстро выполняющийся фрагмент кода. [18]

#### 1.2.4.3 Мьютекс

Мьютекс (mutex — MUTual Exclusion [18]) — механизм взаимного исключения, представляющий собой специальный тип семафора, который реализуется для совместного доступа к ресурсу нескольких задач (общий ресурс).

Чтобы корректно получить доступ к ресурсу, задача предварительно должна попытаться захватить мьютекс (стать его владельцем). При успешном захвате, задача может выполнять работу с общим ресурсом. По завершении работы задача должна освободить мьютекс, чтобы дать возможность работать с общим ресурсом другим задачам-претендентам. Если, при попытке захвата, мьютекс уже захвачен другой задачей, то задача-претендент переводится в заблокированное состояние до тех пор, пока мьютекс не будет освобожден.

#### 1.2.4.4 Критические секции

Критическая секция [20] — это часть программы, которую в один момент времени может выполнять только одна задача или прерывание. Обычно защищаемый критической секцией участок кода начинается с инструкции входа в критическую секцию и заканчивается инструкцией выхода из нее. Данный способ защиты общих ресурсов является очень грубым. Обычно в ОСРВ вход в критическую секцию реализуется запретом всех прерываний, а при выходе — разрешением.

					27.03.04.2023.203.23.05 ВКР	Лист
Изм.	Лист	№ докум.	Подпись	Дата		36



Участки кода, находящиеся внутри критической секции, должны быть как можно короче и выполняться как можно быстрее. Иначе использование критических секций негативно скажется на времени реакции системы на прерывания. Данное требование относится и к реализации обработчиков прерываний без ОСРВ.

### 1.2.5 Программные таймеры

Программный таймер [21] — это инструмент ядра ОСРВ, который можно рассматривать как отдельную задачу, выполняемую в точно заданные моменты времени. При создании программного таймера, ему назначается функция-обработчик и задается период его срабатывания. Управляется таймер планировщиком ОСРВ, поэтому его период не может быть меньше кванта планировщика. Программный таймер можно: запускать, останавливать, сбрасывать, изменять период работы, удалять. Программный таймер может быть интервальным и периодическим.

Интервальный таймер характеризуется тем, что функция-обработчик таймера вызывается только один раз — когда время таймера истечет. Таймер после этого останавливается (переходит в пассивный режим). Такой таймер можно впоследствии запустить заново «вручную». Данный вид программного таймера применяется, когда необходимо организовать однократное выполнение какого-либо действия спустя заданный промежуток времени.

В случае периодического таймера, функция-обработчик таймера вызывается также по завершении счета таймера, но после этого таймер автоматически перезапускается заново, пока его не остановят «вручную».

## 1.3 Стандарт МЭК-61131-3

В силу того, что общепринятого подхода к программированию ПЛК изначально не существовало, поэтому под эгидой организации МЭК (IEC), при участии крупных разработчиков средств автоматизации, было решено сформировать международный стандарт, который стал называться МЭК-61131.

					27.03.04.2023.203.23.05 ВКР	Лист
Изм.	Лист	№ докум.	Подпись	Дата		37

МЭК (IEC, International Electrotechnical Commission, Международная электротехническая комиссия) - международная некоммерческая организация по стандартизации в области электрических, электронных и смежных технологий.

МЭК-61131-3 (IEC-61131-3, ГОСТ Р МЭК-61131-3) - раздел международного стандарта МЭК-61131 (IEC-61131), устанавливающий синтаксис и семантику языков программирования для программируемых устройств управления. Первая редакция стандарта вышла в 1993 году, вторая — в 2003 г., третья — в 2012 г. (IEC 61131-3:2013 / ГОСТ Р МЭК 61131-3-2016) [19].

Стандарт МЭК-61131-3 описывает следующие языки программирования:

- ST (Structured Text) – структурированный текст,
- IL (Instruction List) — список инструкций,
- LD (Ladder Diagram) - релейно-контактные схемы,
- FBD (Function Block Diagram) — функциональные блочные диаграммы,
- SFC (Sequential Function Chart) – последовательные функциональные диаграммы.

### 1.3.1 Язык ST

ST (Structured Text, Структурированный Текст) - это текстовый язык высокого уровня общего назначения, по синтаксису похож на язык Pascal. Удобен для программ с числовым анализом и сложными алгоритмами. Может использоваться в теле функции или функциональных блоков (язык FBD), а также для описания действия и переходов (язык SFC). В понимании этот язык близок программистам среднего и высокого уровней. Фрагмент кода на языке ST приведен на рисунке 1.12. [19] [29]

```
1 IF Ex = TRUE THEN
2   ScaleA:= (Ka*V)+Kb;
3 ELSE
4   ScaleA:= Y;
5 END_IF;
6
```

Рисунок 1.12 - Фрагмент кода на языке ST

### 1.3.2 Язык IL

IL (Instruction List, Список Инструкций) — это текстовый язык низкого уровня, который похож на язык Assembler, но к конкретной процессорной архитектуре не привязан. Был добавлен в стандарт МЭК-61131 для сохранения переносимости кода, реализованного на Assembler, но в 3-й редакции стандарта был отмечен как сложный и неудобный для понимания и помечен как нерекommenдoванный (исключен в некоторых средах программирования). Фрагмент кода на языке IL приведен на рисунке 1.13. [19] [21]

```
1 (* Y(x) = A*x + B *)
2 LD   x
3 MUL  A
4 ADD  B
5 ST   Y
6 |
```

Рисунок 1.13 - Фрагмент кода на языке IL

### 1.3.3 Язык LD

LD (Ladder Diagram, Ступенчатые Диаграммы) — это графический язык высокого уровня, основанный на принципах многоступенчатых релейно-контактных схем, определяющих электрический поток (от шины питания «+» к шине «-»). Язык описывает простые логические правила (только булевы выражения). Основные элементы логики: левая и правая шины питания, контакты, вертикальные и горизонтальные перемычки (проводники), обмотки (катушки) реле. В понимании данный язык близок специалистам по электротехнике. Фрагмент кода на языке LD приведен на рисунке 1.14. [19] [21]

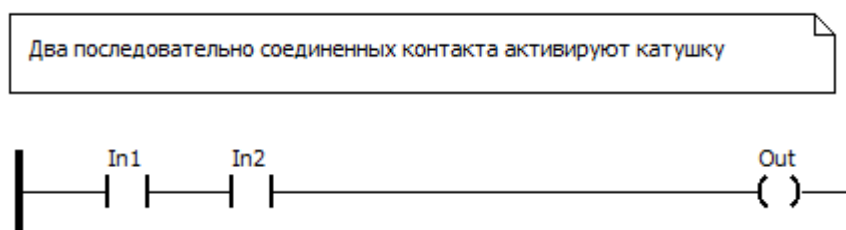


Рисунок 1.14 - Фрагмент кода на языке LD

### 1.3.4 Язык FBD

FBD (Function Block Diagram, Функциональные Блочные Диаграммы) — это графический язык высокого уровня, обеспечивающий управление потоками данных любых типов. Язык обеспечивает объектно-ориентированный подход в программировании, подходит для реализации различных алгоритмов в непрерывных последовательных процессах и визуально похож на реализацию функциональной схемы технологического процесса. Основные элементы логики: переменные, блоки функций и функциональные блоки, соединения и переходы. Язык позволяет использовать большую библиотеку готовых блоков. В понимании данный язык близок специалистам-технологам. Фрагмент кода на языке FBD приведен на рисунке 1.15. [19] [21]

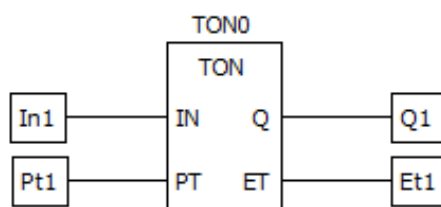


Рисунок 1.15 - Фрагмент кода на языке FBD

### 1.3.5 Язык SFC

SFC (Sequential Function Chart, Последовательность Функциональных Диаграмм) — это графический язык высокого уровня, позволяющий легко описывать последовательное управление процессом, базируясь на системе условий, передающих управление с одной операции на другую. Основные элементы логики: шаги, переходы и прыжки, блоки действий. Язык подходит для реализации различных алгоритмов в непрерывных последовательных процессах. В понимании данный язык близок специалистам-технологам. Фрагмент кода на языке SFC приведен на рисунке 1.16. [19] [21]

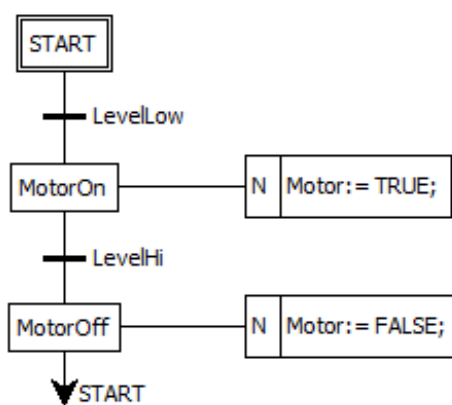


Рисунок 1.16 - Фрагмент кода на языке SFC

### 1.3.6 Среда программирования

Среда программирования стандарта МЭК-61131-3 — это система программных средств, используемая инженерами АСУ ТП для разработки управляющих программ. Программирование в среде, соответственно, выполняется на языках программирования данного стандарта.

Среда программирования, обычно, включает в себя: редакторы языков программирования, транслятор и компилятор программ в машинный код (для конкретного устройства управления и его системы исполнения), программные средства связи с устройством управления (для загрузки и отладки программы), отладчик управляющей программы в режиме реального времени. Некоторые среды программирования предоставляют инструменты для создания «человеко-машинного интерфейса» (HMI).

Некоторые производители средств автоматизации для своих контроллеров разрабатывают собственные среды программирования — это, так называемые, аппаратнозависимые среды. Обычно, эти среды являются закрытыми — не могут быть использованы другими производителями. Например, фирма Siemens только для своих контроллеров разрабатывает собственную среду программирования «Step7».

Также имеются производители, кто занимается только разработкой сред программирования и, как правило, эти среды являются более универсальными - аппаратнонезависимые – могут быть настроены для работы с любым контроллером. Самые известные аппаратнонезависимые среды — Isagraf и CodeSys. Обе эти среды полностью соответствуют стандарту МЭК-61131-3. Помимо самой среды, Isagraf и CodeSys предоставляют собственную встраиваемую систему исполнения, базовая комплектация которой портирована под множество процессорных архитектур (требуется лишь доработка функционала под конкретное устройство управления). Но, эти среды являются «условно бесплатными», то есть накладывают ряд ограничений, которые можно снять, оплатив ту или иную лицензию.

В связи с выше описанными ситуациями, ряд энтузиастов-программистов предприняли попытку разработать аппаратнонезависимую среду программирования, которая соответствовала бы стандарту МЭК-61131-3 и опиралась на стандарт открытого программного обеспечения: открытый исходный код, свободное распространение, отсутствие ограничивающих лицензий. Одной из таких стала - среда программирования Beremiz.

Сравнительный анализ известных сред приведен в таблице 1.1.

Таблица 1.1 - Сравнение сред программирования

Показатель	Step7	Isagraf	CodeSys	Beremiz
Соответствие стандарту МЭК-61131-3	частичное	полное	полное	полное
Ограничения на работу в среде программирования	платная лицензия (ключ)	платная лицензия (ключ)	есть платные библиотеки	нет
Возможность редактирования конфигурации и файлов проекта управляющей программы вне среды	нет	нет	нет	есть

Окончание таблицы 1.1

Показатель	Step7	Isagraf	CodeSys	Beremiz
Возможность модернизации среды программирования под свои нужды	нет	нет	нет	есть ( <i>OpenSource, Python, XML</i> )
Ограничения на систему исполнения для устройства управления	платная лицензия (ключ)	платная лицензия (ключ)	платная лицензия (ключ)	нет (любой процессор можно «превратить» в ПЛК)

## 1.4 Среда программирования Beremiz

Beremiz - это среда программирования и отладки управляющих программ для программируемых устройств управления, соответствует стандарту МЭК-61131-3 и опирается на открытые стандарты: [20]

- программное обеспечение открыто (open-source) и распространяется с исходными кодами (среда написана на языке Python);
- гибкость в изменении существующих и добавлении новых компонентов:
  - на языке Python (интерфейс и функционал среды программирования),
  - на языке XML (файлы-описания систем исполнения, плагинов и т. п.);
- нет ограничений и зависимостей для аппаратных платформ — любой процессор (в том числе обычный персональный компьютер) можно «превратить» в программируемое устройство.

Организации, использующие Beremiz в своих разработках: [20][25]

- Smarteh (Словения) в качестве среды программирования и исполнения для собственных устройств (программируемые реле, ПЛК, сенсорные программируемые контроллеры — СПК).
- KOSMOS, совместно с институтом электронных технологий Южной Кореи (Южная Корея) в качестве среды программирования и исполнения для собственных систем управления приводами.

- ООО НПК «Нуклерон» (г.Пермь) в качестве среды программирования и исполнения для собственных программируемых реле NUC на базе микроконтроллеров STM32 (ARM Cortex-M3, Cortex-M4); разрабатывают собственную модернизированную версию Veremiz (YAPLC-IDE) и систему исполнения (целевую систему YAPLC-RTE), исходные коды которых находятся в открытом доступе.

- ПАО «ИНЭУМ им. И.С. Брука» в качестве среды программирования и исполнения для собственных устройств (например, ПЛК серии CM1820M на базе отечественных микропроцессоров «Эльбрус» и SPARC, контроллера сбора данных КМАВ-С на базе микропроцессора Миландр К1986\*, а также линейек одноплатных ПЛК и сетевых коммутаторов на микроконтроллерах ARM и процессорах x86).

- ПАО «НефтеАвтоматика» (г.Уфа) в качестве среды программирования и исполнения для собственных устройств (например, ПЛК MKLogic на базе микроконтроллера архитектуры ARM Cortex-M4).

- ООО «НГП Информ» в качестве среды программирования и исполнения для собственных устройств (например, ПЛК CILk PAC на базе микроконтроллеров ARM Cortex-M3 и Cortex-A8).

- ООО «Информационные технологии» / ЗАО «Ламинарные системы» (г.Миасс) в качестве среды программирования и исполнения для собственных устройств (например, ПЛК IT-2004 на базе микроконтроллера STM32 – ARM Cortex-M3).

Пример главного окна среды Veremiz приведено на рисунке 1.17.



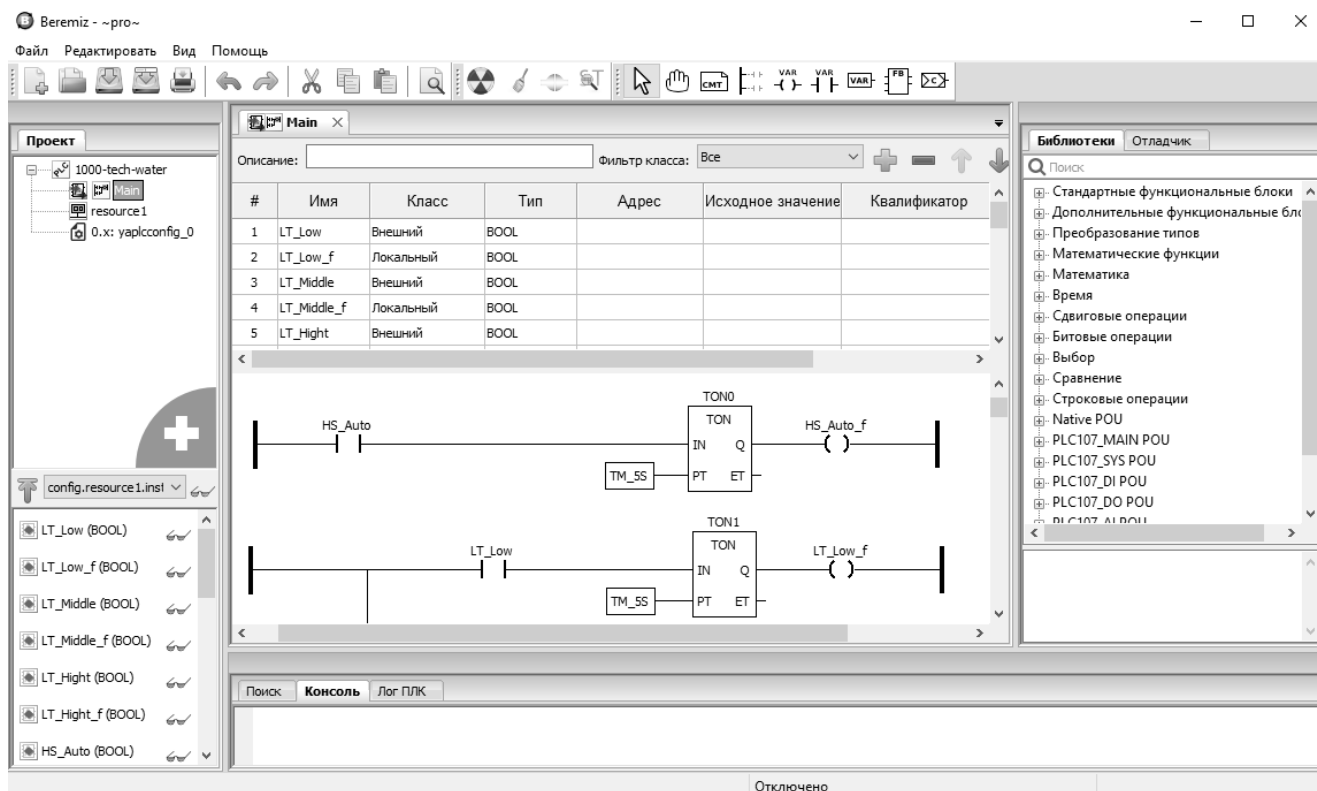


Рисунок 1.17 - Главное окно пользовательского интерфейса среды Beremiz

### 1.4.1 Beremiz YAPLC

Так как среда программирования Beremiz является аппаратно-независимой, соответственно, она изначально не имеет инструментов для работы с конкретным контроллером (нет компилятора и компоновщика, нет модуля связи, нет загрузчика, нет описания вызовов системы исполнения). Поддержка подобных инструментов здесь полностью отдана разработчику контроллера (и его системы исполнения) и реализуется с помощью специальных файлов-модулей, именуемых как «Описание целевой системы исполнения» и интегрируемых непосредственно в среду Beremiz. Пример реализации подобных модулей можно посмотреть в проекте Beremiz YAPLC (компания ООО НПК «Нуклерон», г.Пермь, исходные коды открыты и доступны в сети Интернет). [26]

В Beremiz YAPLC интегрированы следующие модули:

- модуль описания целевых платформ (targets), включая:
  - карта адресов регистров данных,
  - дополнительный код на языке C для взаимодействия управляющей программы с системой исполнения целевой платформы,

- настройки кросс-компилирования,
- инструменты кросс-компиляции и загрузки управляющей программы;
- модуль связи с целевым устройством через последовательный интерфейс (yaplcserial) для отладки в режиме реального времени.

В Beremiz YAPLC описание целевой системы находится в директории IDE/yaplc targets, куда входят: [26]

- директория <имя\_целевой\_платформы> следующего содержания:
  - \_\_init\_\_.py, XSD — файл-модуль интеграции описания целевой платформы в среду программирования (файл в нотации языка Python),
  - extentions.cfg – файл-карта адресов регистров данных целевой платформы,
  - plc\_<имя\_целевого\_устройства>\_main.c – файл дополнительного кода на языке C, специфичный для целевой платформы, который автоматически будет добавлен в конец файла plc\_main.c в процессе компиляции управляющей программы.
- toolchain\_yaplc.py — файл-модуль интеграции кросс-компилятора в среду программирования.
- toolchain\_yaplc\_stm32.py — файл-модуль интеграции программы-загрузчика (stm32flash для микроконтроллеров STM32) в среду программирования.

Дополнительные исходные файлы для целевой системы располагаются в директории RTE/src/bsp/<имя\_целевого\_устройства>, где:

- target-app.ld — файл-скрипт компоновщика компилятора,
- plc\_abi.h – файл с описанием служебных структур, используемых для взаимодействия управляющей программы и системы исполнения.

#### 1.4.2 Схема создания и загрузки управляющей программы

Проект управляющей программы в среде Beremiz представлен в формате XML и хранится в отдельной директории, куда изначально входят файлы:

- beremiz.xml – описание целевого устройства (тип и настройки связи),
- <название проекта>.xml – файл проекта (исходный код проекта).

Для Beremiz характерна следующая схема создания управляющей программы (рисунок 1.18): [20][21]

1) Создается новый проект:

- а) задается имя проекта и место его расположения;
- б) настраивается проект (выбирает устройство управления);
- в) определяются глобальные переменные;
- г) реализуется алгоритм управления на языках программирования;
- д) определяются настройки задач для исполнения программ;

2) Выполняется первый этап компиляции - проект передается компилятору MatIEC (поставляется со средой программирования Beremiz), который:

- а) формирует промежуточный код на языке ST и сохраняет его в файл <название проекта>.st (в директории build), куда входит:
  - исходный код на языках IL, LD, FBD, преобразованный в единый код на языке ST;
  - исходный код на языке ST;
  - дополнительный код логики работы задач и ресурсов проекта.
- б) преобразует промежуточный код на языках ST в код на языке ANSI C и сохраняет результат в следующие файлы (в директории build):
  - config.h, config.c, resource.c POUS.h, POUS.c – код реализации алгоритмов, логики работы программных модулей и ресурсов проекта;
  - plc\_main.c – дополнительный код с функциями управления программой (например, разовое исполнение, останов, тактирование рабочего цикла и т.п.), а также код, специфичный для целевого устройства (архитектуры);
  - plc\_debugger.c – код с функционалом отладки управляющей программы на целевом устройстве из среды программирования Beremiz.

3) Выполняется второй этап компиляции - сформированный код на языке ANSI C передается кросс-компилятору, соответствующего архитектуре выбранного устройства управления, который формирует машинный код и сохраняет его в файлах \*.elf, \*.hex, \*.bin (зависит от настроек компилятора).

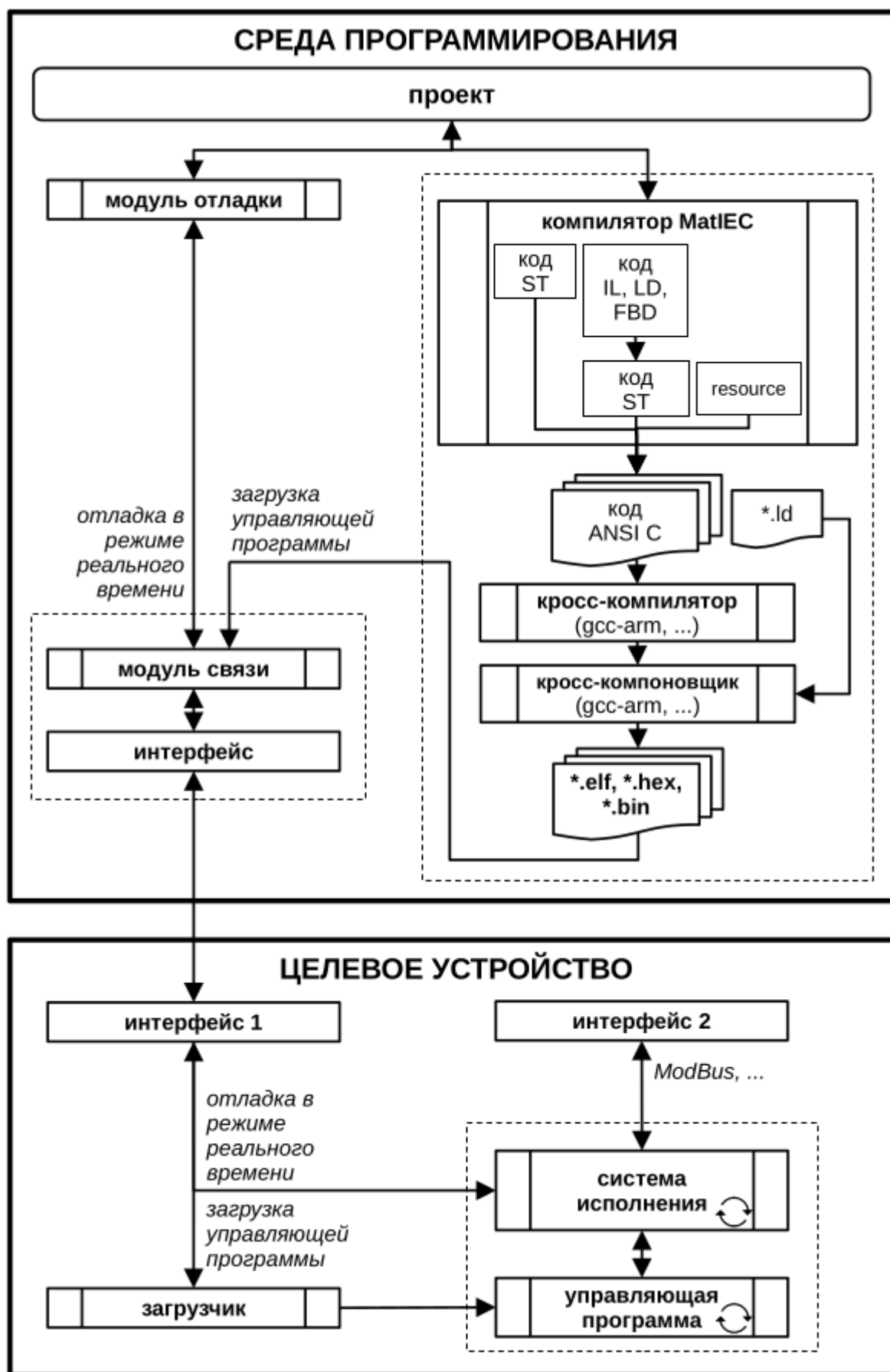


Рисунок 1.18 - Схема создания и загрузки управляющей программы

4) Сформированный исполняемый файл управляющей программы загружается в память устройства управления через специальный интерфейс передачи данных (например, USB, RS-485, ETHERNET) и с помощью специальной программы (загрузчик / программатор); загрузчик может быть как отдельной программой, запускаемой самостоятельно или из среды программирования, либо модулем самой среды программирования (Beremiz YAPLC предоставляет примеры и тех и других решений).

5) Система исполнения устройства управления (обычно, после перезагрузки) начинает исполнять загруженную управляющую программу.

### 1.4.3 Связь системы исполнения и управляющей программы

Среда Beremiz YAPLC предоставляет следующую схему связи (взаимодействия) системы исполнения ПЛК и загруженной в него управляющей программы (рисунок 1.19): [21][25][26]

1) В памяти программ устройства управления выделяется раздел для хранения управляющей программы (характеристики раздела известны: размер и стартовый адрес).

2) Характеристики раздела памяти для управляющей программы описываются в файле компоновщика target-app.ld (секция MEMORY).

3) В файле plc\_<имя\_целевого\_устройства>\_main.c определяется таблица указателей plc\_yaplc\_app (экземпляр структуры типа plc\_app\_abi\_t, определенной в файле plc\_abi.h), которая содержит:

- контрольные суммы (.id, .check\_id),
- номер версии системы исполнения (.rte\_ver),
- минимальное значение периода исполнения управляющей программы в наносекундах (.common\_ticktime),
- указатель на функцию исполнения одного цикла управляющей программы (.run),
- и прочие данные.

4) Таблица указателей plc\_yaplc\_app размещается в самом начале управляющей программы (чтобы адрес начала таблицы соответствовал стартовому адресу раздела), для этого:

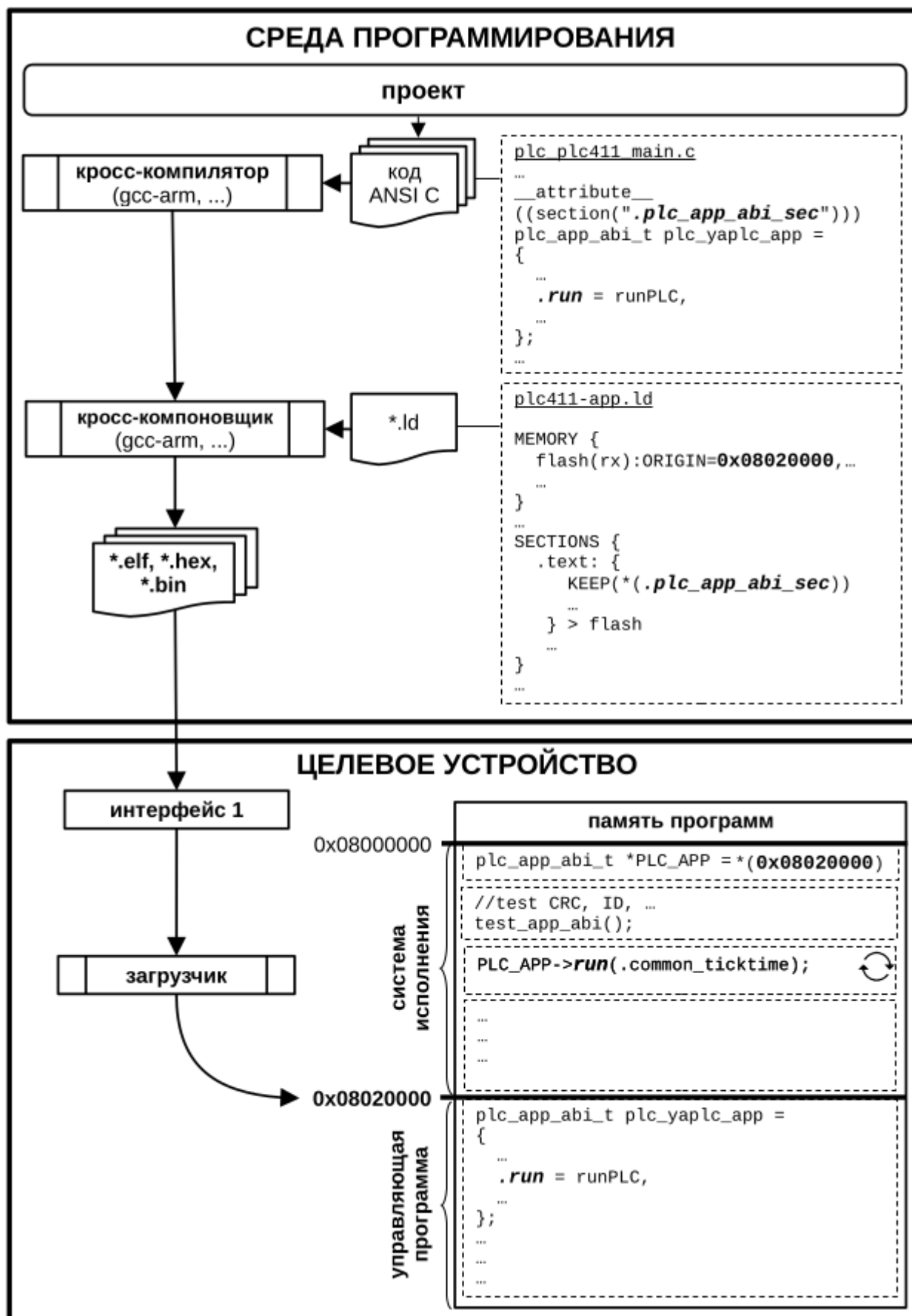


Рисунок 1.19 - Схема связи системы исполнения и управляющей программы

- в файле `plc_plc411_main.c` перед определением таблицы `plc_uarplc_app` принудительно указывается адрес в секции памяти `.plc_app_abi_sec`;
- в файле компоновщика `target-app.ld` вначале секции `.text` размещается секция `.plc_app_abi_sec`.

5) В коде системы исполнения определяется переменная `PLC_APP`, указывающая на таблицу `plc_uarplc_app` (указатель на стартовый адрес памяти устройства управления - на раздел для хранения управляющей программы).

6) После компиляции и компоновки проекта в среде программирования, управляющая программа загружается в устройство управления — в выделенный раздел памяти.

7) Устройство управления перезагружается.

8) После загрузки устройства управления, на первом цикле выполнения системы исполнения выполняется проверка данных в переменной `PLC_APP`: наличие контрольной суммы, идентификаторов и прочие данные - загружена ли управляющая программа и корректна ли она.

9) Если результат проверки таблицы указателей является успешным, то можно исполнять управляющую программу, вызывая метод `PLC_APP->run()` по прерыванию или циклически (время цикла в наносекундах доступно в опции `PLC_APP->common_ticktime`).

#### 1.4.4 Связь переменных с регистрами данных

Связывание переменных проекта управляющей программы с регистрами данных системы исполнения (значение и настройки канала В/В, системные данные и т.п.) осуществляется в среде программирования *Veremiz* путем задания соответствующего адреса в редакторе констант и переменных (рисунок 1.20). [21]

Адрес задается вручную или выбирается из специального диалогового окна (рисунок 1.21) [21]. Список адресов для этого диалогового окна формируется из файла `extentions.cfg` (карта адресов каналов В/В из описания системы исполнения целевого устройства).

#	Имя	Класс	Тип	Адрес	Исходное значение	Квалификатор	Описание
1	G_MCU_TEMP	Глобальный	DWORD	%MD7.4.0	0.0		
2	G_PT0_TEMP	Глобальный	REAL		0.0		
3	G_DO0_PWM_EN	Глобальный	BOOL		TRUE		
4	G_DO0_PWM_DU	Глобальный	REAL		0.0		
5	G_INITED	Глобальный	BOOL		FALSE		

Рисунок 1.20 - Связь переменной с регистром данных (пример)

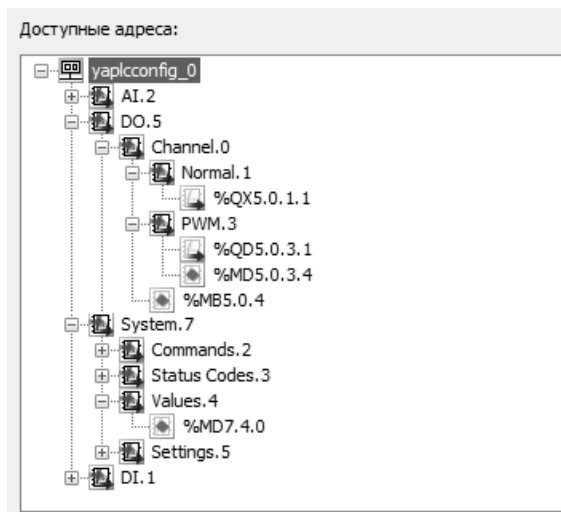


Рисунок 1.21 - Диалог выбора адреса регистра данных (пример)

Пример формата адреса регистра данных приведен в таблице 1.2.

Таблица 1.2 - Формат адреса регистра данных

зона	тип данных	группа	канал В/В	подгруппа	регистр
Q	X	5	0	1	1
M	D	7		4	0

Зона - символьный код вида: I –вход, Q – выход, M – память.

Тип данных — символьный код (см. таблицу 1.3). [21]

Группа — числовой код группы регистров (каналы В/В, системные и пользовательские данные).

Канал В/В — числовой код канала В/В в пределах Группы. Этот код применяется только для адресации регистров, связанных с каналами В/В (может отсутствовать в составе адреса).



Подгруппа — числовой код подгруппы регистров в пределах одного Канала В/В или Группы. Этот код применяется, если в пределах Группы или Канала В/В имеются регистры, разделенные на дополнительные подгруппы (например, для каналов DO - подгруппы по режимам работы, для пользовательских данных — подгруппы по типам данных).

Регистр — числовой код регистра в пределах Подгруппы.

Таблица 1.3 - Коды типов данных

Код	Тип		Диапазон значений	Размер	
	IEC	Си		биты	байты
X	BOOL	uint8	0 ... 255 (0, 1)	8	1
B	BYTE USINT SINT	uint8 int8	0 ... 255 -128 ... 127	8	1
W	WORD UINT INT	uint16 int16	0 ... 65535 -32768 ... 32767	16	2
D	DWORD UDINT DINT REAL	uint32 int32 float	0 ... 4294967295 -2147483648 ... 2147483647 $3.4 \cdot 10^{-38} \dots 3.4 \cdot 10^{38}$	32	4
L	LWORD ULINT LINT LREAL	uint64 int64 double	0 ... $(2^{64}-1)$ $-(2^{63}-1) \dots (2^{63}-1)$ $1.7 \cdot 10^{-308} \dots 1.7 \cdot 10^{308}$	64	8

Например, адрес %MD7.4.0 – «Температура ПЛК, °C», где:

- M – код класса «Память»,
- D – код типа REAL (из руководства на целевую платформу),
- код канала В/В отсутствует (это системный регистр, не канал В/В)
- 7 – код группы «Системные регистры»,
- 4 – код подгруппы «Значения/Показания»,
- 0 – код регистра в пределах подгруппы.

## 2 ОСНОВНОЙ РАЗДЕЛ

### 2.1 Функциональная схема «ПЛК411»

Заданием определено устройство управления класса «монолитный ПЛК» с кодовым обозначением «ПЛК411»:

- аппаратная платформа в рамках данной работы не разрабатывается;
- функциональная схема устройства приведена в ПРИЛОЖЕНИИ Б;
- внешний вид корпуса устройства приведен на рисунке 2.1;
- общие технические характеристики приведены в таблице 2.1.

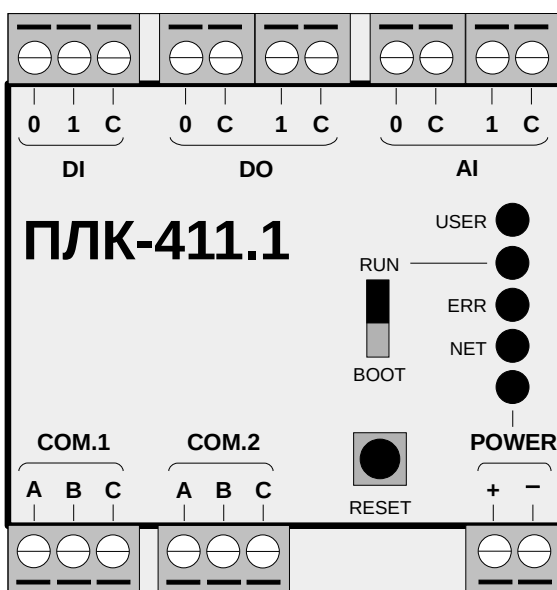


Рисунок 2.1 - Внешний вид корпуса «ПЛК411»

#### 2.1.1 Технические характеристики ПЛК

Таблица 2.1 - Общие технические характеристики «ПЛК411»

Характеристика	Значение
Микроконтроллер	STM32F411CE
DI тип	Каналы дискретного ввода
количество	= гальваническая развязка, общий вывод (C)
функции	= 2 шт. (0, 1, C - общий)
электрический сигнал	= нормальный, счетчик импульсов, тахометр, инкрементальный энкодер
	= 0 - 50 В постоянного тока, до 5 мА
	0 - 4 В - логический «0», 10 - 50 В - логическая «1»

Продолжение таблицы 2.1

Характеристика	Значение
DO тип  количество функции электрический сигнал	Каналы дискретного вывода = транзисторный с открытым истоком, гальваническая развязка = 2 шт. (0, 0.C, 1, 1.C) = нормальный, нормальный быстрый, ШИМ ( $T_{min}=100$ кГц) = 10 - 50 В постоянного тока, до 1 А непрерывно, до 5 А в импульсе (300 мкс, скважность 50%)
AI количество функции электрический сигнал разрядность вх.сопротивление	Каналы аналогового ввода = 2 шт. (0, 0.C, 1, 1.C) = опрос ( $T=100$ мсек), калибровка = 0 - 10 В постоянного тока, до 5 мА = 12 бит (точность $\pm 10$ мВ) = 100 кОм
COM.1 тип количество функции  сетевые настройки	Последовательный сетевой интерфейс = RS-485, гальваническая развязка = 1 шт. (А, В, С – общий) = обновление системы исполнения и управляющей программы, отладка управляющей программы из среды Veremiz в режиме реального времени = 115200 8 N 1, сетевой адрес: 1
COM.2 тип количество	Последовательный сетевой интерфейс = RS-485, гальваническая развязка = 1 шт. (А, В, С – общий) = доступ к регистрам данных по протоколу ModBus RTU (режим Slave, чтение/запись) = 115200 8 N 1, сетевой адрес: 1
POWER электрический сигнал	Питание = 9 - 36 В постоянного тока, не более 5 Вт
RUN / BOOT тип функции	Переключатель = с фиксацией положения = выбор режима работы системы исполнения RUN – работа BOOT – обновление программного обеспечения

## Окончание таблицы 2.1

Характеристика	Значение
RESET тип функции	Кнопка = без фиксации состояния = перезагрузка устройства
USER функции	Светодиодный индикатор = пользовательская индикация
RUN функции	Светодиодный индикатор = индикация работы устройства в режиме RUN: постоянно горит - работает система исполнения мигает - работает система исполнения и управляющая программа
ERR функции	Светодиодный индикатор = индикация наличия ошибок при работе системы исполнения или при активации безопасного состояния выходов
NET функции	Светодиодный индикатор = индикация передачи данных по сети мигает при передаче данных по одному из сетевых интерфейсов (COM.1 и/или COM.2)
POWER функции	Светодиодный индикатор = индикация наличия питания постоянно горит при наличии питания

Весь заявленный функционал (включая исполнение управляющей программы) обеспечивается системой исполнения «ПЛК411».

Все настройки, состояния и команды размещаются в памяти ПЛК - в таблицах регистров данных, сгруппированных по типу и правам доступа (чтение / запись). Доступ к этим регистрам осуществляется как по сети через интерфейс COM.2 (протокол ModBus RTU), так и из управляющей программы через адресуемые переменные.

В библиотеке функций среды Veremiz (для системы исполнения «ПЛК411») доступны специальные функции и функциональные блоки, обеспечивающие еще один способ доступа к регистрам данных ПЛК, используя соответствующие системные вызовы.

## 2.1.2 Технические характеристики микроконтроллера

Основой «ПЛК411» является микроконтроллер STM32F411CE производства фирмы STMicroelectronics. Данный микроконтроллер построен на ядре с архитектурой ARM Cortex-M4. Основные технические характеристики микроконтроллера приведены в таблице 2.2. [32] [33]

Таблица 2.2 - Технические характеристики микроконтроллера STM32F411CE

Характеристика	Значение
Ядро	ARM Cortex-M4
разрядность	= 32-бит
архитектура	= RISC
дополнительно	+ аппаратная поддержка чисел с плавающей точкой
Тактовая частота	до 100 МГц + встроенный резонатор 16 МГц («ПЛК411» работает от внешнего резонатора 25 МГц)
Память программ	FLASH
размер	= 512 кБайт
ОЗУ	SRAM
размер	= 128 кБайт
Порты В/В	36
Контроллер прямого доступа к памяти	1x DMA (16 каналов)
АЦП	1x ADC (12-бит, 16 каналов) + встроенный температурный датчик
ЦАП	нет
Таймеры	7x 16-бит (1 — системный) 2x 32-бит
Коммуникационные интерфейсы	1x SPI 3x I2C 3x USART 1x SDIO 1x USB OTG 1x JTAG & SWD
Часы реального времени	1x RTC + встроенный резонатор 32,768 кГц
Встроенный системный загрузчик	Есть (загрузка программ через UART)

## Окончание таблицы 2.2

Характеристика	Значение
Сторожевой таймер	2х
Электропитание	1,7 — 3,6 В постоянного тока
Рабочие температуры	-40 +85 °С
Тип корпуса	UFQFPN48

### 2.1.3 Распределение памяти

Микроконтроллер STM32F411CE располагает встроенной памятью: ОЗУ (SRAM) и память программ (FLASH). Предлагается следующая схема ее распределения (таблица 2.3):

Таблица 2.3 - Распределение памяти микроконтроллера STM32F411CE

Память	Раздел		
	имя (назначение)	размер	начальный адрес
ОЗУ	RAM (система исполнения)	100 КБ	0x20000000
	RAM_APP (управляющая программа)	28 КБ	0x20019000
Память программ	FLASH (система исполнения)	256 КБ	0x08000000
	FLASH_APP (система исполнения)	256 КБ	0x08040000

Для связи управляющей программы и системы исполнения необходимо разместить таблицу указателей `plc_yaplc_app` в самом начале раздела FLASH\_APP - адрес 0x08040000 (см. п. 1.4.3 «Связь системы исполнения и управляющей программы»). Этот адрес необходимо указать в исходном коде системы исполнения и дополнительном исходном коде описания целевой системы исполнения для среды Veremiz.

### 2.1.4 Режим работы ПЛК

Подразумевается работа ПЛК в следующих режимах:

- RESET – Перезагрузка,

- BOOT – Обновление встроенного программного обеспечения,
- RUN – Работа.

#### 2.1.4.1 Режим «Перезагрузка»

В этом режиме выполняется аппаратная перезагрузка микроконтроллера ПЛК. На первом цикле работы (после перезагрузки) микроконтроллер ПЛК на аппаратном уровне (по состоянию переключателя «RUN/BOOT») определяет в каком режиме он дальше будет работать.

Перезагрузка выполняется следующим образом:

- 1) нажать кнопку «RESET » (на лицевой панели ПЛК).

#### 2.1.4.2 Режим «Обновление встроенного программного обеспечения»

В этом режиме запускается системный загрузчик микроконтроллера ПЛК, который предоставляет функционал (протокол) обновления встроенного ПО: системы исполнения или управляющей программы. С помощью специального устройства и программного обеспечения (программатор) системному загрузчику микроконтроллера через интерфейс COM.1 передается машинный код с указанием по какому адресу в памяти программ его записывать.

Активация режима выполняется следующим образом:

- 1) перевести переключатель «RUN/BOOT» в положение «BOOT»;
- 2) перезагрузить ПЛК.

#### 2.1.4.3 Режим «Работа»

В этом режиме запускается система исполнения ПЛК.

Активация режима выполняется следующим образом:

- 1) перевести переключатель «RUN/BOOT» в положение «RUN»;
- 2) перезагрузить ПЛК.

### 2.1.5 Каналы «DI»

Дискретный вход (DI, Digital Input) нужен для ввода в ПЛК электрических сигналов, ассоциированными только с двумя состояниями: включено или отключено, есть сигнал или нет сигнала. Эти состояния задаются уровнем напряжения, которое подается на DI. Далее будет дано описание функций, настроек и контрольных состояний каналов DI, распределенных по регистрам данных ПЛК.

#### 2.1.5.1 Общие настройки

Регистр «DI.x: Режим работы» - режим работы канала:

- = 0 — выключен,
- = 1 — нормальный,
- = 1 — счетчик импульсов,
- = 2 — тахометр (частотомер),
- = 3 – инкрементальный энкодер (счетчик),
- = 4 – инкрементальный энкодер (счетчик и тахометр).

Режимы 3 и 4 доступны только для входов 0 и 2, так как входы здесь работают попарно. Например, при выборе режима «3» для входа DI.0, автоматически режим «3» назначается и на вход «DI.1» - эти два входа будут работать в паре.

#### 2.1.5.2 Режим «Выключен»

В этом режиме вход DI.x не работает — уровни с физического канала ввода не принимаются. Значение уровня доступно в регистре «DI.x нормальный: Значение» и автоматически приравнивается логическому «0».

#### 2.1.5.3 Режим «Нормальный»

В этом режиме вход DI.x работает как обычный дискретный: уровни «0» (низкий) или «1» (высокий). Значение уровня доступно в регистре «DI.x нормальный: Значение». Изменение уровня на физическом канале ввода

					27.03.04.2023.203.23.05 ВКР	Лист
Изм.	Лист	№ докум.	Подпись	Дата		60



детектируется (по смене фронта сигнала) аппаратным прерыванием микроконтроллера ПЛК.

Регистр «DI.x нормальный: Значение» - значение уровня:

= FALSE — низкий уровень (по-умолчанию),

= TRUE — высокий уровень

#### 2.1.5.4 Режим «Счетчик импульсов»

В этом режиме вход DI.x работает на счет количества поступающих импульсов (по переднему фронту). Значение счетчика записывается в регистр «DIx счетчик: Значением (имп)».

Сброс значения счетчика:

- 1) записать логическую «1» в регистр «DI.x: Команда сброса счетчика».

После обнуления счетчика, команда автоматически сбрасывается в «0» и счет начинается сначала (с нуля). Значение счетчика также автоматически сбрасывается при смене режима работы входа DI.x.

Для данного режима доступен алгоритм работы счетчика по уставке.

Настройка входа DI.x на работу по уставке:

- 1) записать уставку в регистр «DI.x счетчик: Уставка (имп)»;
- 2) разрешить работу по уставке, записав логическую «1» в регистр «DI.x счетчик: Разрешение работы по уставке».

Теперь, если значение счетного регистра будет равно уставке, то в регистр «DI.x счетчик: Уставка достигнута, признак» запишется логическая «1», иначе – «0». Сброс признака выполняется командой сброса счетчика.

Регистр «DI.x: Значение (имп)» - значение счетчика:

Регистр «DI.x: Уставка (имп)» - значение уставки для счетчика:

= 0 ... 4294967295.

Регистр «DI.x: Уставка достигнута, флаг» - логический признак достижения уставки счетчиком:

- = FALSE — счетчик не равен уставке,
- = TRUE — счетчик равен уставке.

Регистр «DI.x: Разрешение уставки» - логическая настройка, дающая разрешение на работу счетчика по уставке:

- = FALSE — работа по уставке исключена,
- = TRUE — работа по уставке включена.

#### 2.1.5.5 Режим «Тахометр»

В этом режиме вход DI.x работает на счет количества поступающих импульсов в секунду. Значение счетчика записывается в регистр «DI.x тахометр: Значение (имп/сек)». Обновление регистра выполняется каждые 100 мсек.

Обнуления счетчика для этого режима не предусмотрено (сбрасывается автоматически при пропадании импульсов на входе).

Для режима «Тахометр» доступен алгоритм работы счетчика по уставке.

Настройка входа на работу по уставке:

- 1) записать уставку в регистр «DI.x тахометр: Уставка (имп/сек)»;
- 2) разрешить работу по уставке, записав логическую «1» в регистр «DI.x тахометр: Разрешение работы по уставке».

Теперь, если значение счетного регистра будет равно уставке, то в регистр «DI.x тахометр: Уставка достигнута, признак» запишется логическая «1», иначе – «0». Сброс признака выполняется командой сброса счетчика.

Регистр «DI.x: Значение (имп/сек)» значение счетчика тахометра:

Регистр «DI.x: Уставка (имп/сек)» - уставка для счетчика тахометра:

= 0 ... 65535.

Регистр «DI.x: Уставка достигнута, флаг» - логический признак достижения уставки счетчиком тахометра:

- = FALSE — счетчик не равен уставке,
- = TRUE — счетчик равен уставке.

Регистр «DI.x: Разрешение уставки» - логическая настройка, дающая разрешение на работу счетчика тахометра по уставке:

- = FALSE — работа по уставке выключена,
- = TRUE — работа по уставке включена.

#### 2.1.5.6 Режим «Инкрементальный энкодер (счетчик)»

Режим доступен только для входа DI.0, так как входы здесь работают попарно: DI.0 (первичный) — DI.1 (вторичный). Режим применяется ко всем входам пары. Например, задали режим «4» для входа DI.0 - режим входа DI.1 автоматически устанавливается также в «4». Например, если в данном режиме к первичному входу пары подключить фазу «А» аппаратного энкодера, а ко вторичному входу пары - фазу «В», то при вращении вала энкодера по часовой стрелке будет инкрементироваться значение счетного регистра «DI.x счетчик: Значение (имп)» первичного входа, а при вращении против часовой стрелки будет инкрементироваться значение аналогичного регистра вторичного входа. Соответственно, при смене подключения фаз поменяются и номера счетных регистров.

При смене режима:

- 1) для первичного входа пары применяется заданный режим,
- 2) для вторичного входа пары автоматически применяется режим «0» (Нормальный).

Обнуление счетного регистра:

- 1) подать логическую «1» в регистр «DI.x: Сброс счетчика, команда» первичного входа пары.

Обнуляются все счетчики пары. После обнуления, команда автоматически сбрасывается в «0» и счет начинается сначала (с нуля). Также счетчик автоматически сбрасывается при смене режима работы входа.

#### 2.1.5.7 Режим «Инкрементальный энкодер (счетчик и тахометр)»

Режим доступен только для входа DI.0, так как входы здесь работают попарно: DI.0 (первичный) — DI.1 (вторичный).

Режим применяется ко всем входам пары. Например, задали режим «5» для входа DI.0 - режим входа DI.1 автоматически устанавливается также в «5». Например, если в данном режиме к первичному входу пары подключить фазу «А» аппаратного энкодера, а ко вторичному входу пары - фазу «В», то при вращении вала энкодера по часовой стрелке будет инкрементироваться значение счетного регистра «DI.x счетчик: Значение (имп)» первичного входа, а при вращении против часовой стрелки будет инкрементироваться значение вторичного входа. Соответственно, при смене подключения фаз поменяются и номера счетных регистров. Кроме того, в регистр «DI.x тахометр: Значение (имп/сек)» вторичного входа пары будет записываться значение импульсов в секунду вне зависимости от направления вращения вала энкодера.

При смене режима:

- 1) для первичного входа пары применяется заданный режим,
- 2) для вторичного входа пары автоматически применяется режим «0» (Нормальный).

Обнуление счетного регистра:

- 1) подать логическую «1» в регистр «DI.x: Сброс счетчика, команда» первичного входа пары.

Обнуляются все счетчики пары. После обнуления, команда автоматически сбрасывается в «0» и счет начинается сначала (с нуля). Также счетчик автоматически сбрасывается при смене режима работы входа.

#### 2.1.5.8 Сброс счетчиков

Регистр «DI.x: Сброс счетчика, команда» - сбрасывает значение счетчика:

- = 0 — не сбрасывать,
- = 1 — сбросить.

#### 2.1.6 Каналы «DO»

Дискретный выход (DO, Digital Output) нужен для вывода из ПЛК электрических сигналов, ассоциированными только с двумя состояниями: включено или отключено, есть сигнал или нет сигнала. Эти состояния задаются уровнем напряжения, которое подается на DO. Далее будет дано описание функций, настроек и контрольных состояний каналов DO, распределенных по регистрам данных ПЛК.

##### 2.1.6.1 Общие настройки

Регистр «DO.x: Режим работы»:

- = 0 — выключен,
- = 1 – нормальный,
- = 2 - быстрый,
- = 3 – ШИМ.

##### 2.1.6.2 Режим «Выключен»

В этом режиме выход DO.x не работает — уровни на физический канал вывода не поступают.

##### 2.1.6.3 Режим «Нормальный»

В этом режиме выход DO.x работает как обычный дискретный: уровни «0» (низкий) или «1» (высокий). Для установки уровня предназначен регистр «DO.x нормальный: Значение». Установленный в регистре уровень автоматически передается на соответствующий физический канал вывода ПЛК.

Регистр «DO.x: Значение» - логический уровень нормального дискретного выхода:

- = FALSE — низкий уровень (по-умолчанию),
- = TRUE — высокий уровень.

#### 2.1.6.4 Режим «Быстрый»

В этом режиме выход DO.x работает по аналогии с режимом «Нормальный», но:

- для режима «Нормальный»:
  - измененное в пользовательской программе значение регистра будет передано на физический вывод только после завершения рабочего цикла управляющей программы;
- для режима «Быстрый»:
  - измененное в управляющей программе значение регистра будет передано на физический вывод сразу же, не дожидаясь завершения рабочего цикла управляющей программы.

Регистр «DO.x: Значение» - логический уровень быстрого выхода:

- = FALSE — низкий уровень (по-умолчанию),
- = TRUE — высокий уровень.

#### 2.1.6.5 Режим «ШИМ»

В этом режиме выход DO.x работает в режиме ШИМ: на канал физического вывода автоматически выдаются импульс заданной длительности и периодичности. Чтобы не нагружать вычислительные мощности микроконтроллера, ШИМ для канала вывода реализуется встроенными аппаратными таймерами (32-битные,  $T_{\min}=100$  кГц).

ШИМ поканально настраивается через следующие регистры:

- «DO.x ШИМ: Работа (% от периода)»,
- «DO.x ШИМ: Период (миллисекунды)»,
- «DO.x ШИМ: Разрешение работы».

Включение ШИМ:

- 1) записать логическую «1» в регистр «DO.x ШИМ: Разрешение работы».

Выключение ШИМ (на выходе постоянный логический «0»):

- 1) записать логический «0» в регистр «DO.x ШИМ: Разрешение работы».

Регистр «DO.x: Работа (% от периода)» - коэффициент заполнения - длительность в % от периода:

= 0.0 ... 100.0

= 0.0 — нет заполнения — на выходе логический «0»,

= 100.0 — полное заполнение — на выходе логическая «1».

Время длительности импульса (в единицах времени) вычисляется по следующей формуле:

$$t = \left( \frac{T}{100} \right) \cdot D \quad (1)$$

где,  $t$  — время длительности импульса ШИМ (миллисекунды),

$T$  — период ШИМ (миллисекунды),

$D$  — коэффициент заполнения ШИМ (% от периода).

Регистр «DO.x: Разрешение работы» - включение/выключение ШИМ:

= FALSE — выключить,

= TRUE — включить.

Регистр «DO.x: Период (миллисекунды)» - период ШИМ:

= 100 ... 4294967295

= < 100 — соответствует периоду в 0 миллисекунд (на выходе будет логический «0»).

### 2.1.7 Каналы «AI»

Аналоговый ввод (AI, Analog Input) нужен для ввод в ПЛК аналоговых сигналов (температура, давление и других физических величин), которые измеряются соответствующими датчиками. Эти сигналы задаются уровнем

					27.03.04.2023.203.23.05 ВКР	Лист
Изм.	Лист	№ докум.	Подпись	Дата		67

напряжения, которое подается на AI. Далее будет дано описание функций, настроек и контрольных состояний каналов AI, распределенных по регистрам данных ПЛК.

#### 2.1.7.1 Общие настройки

Регистр «AI.x: Режим работы»:

- = 0 — выключен,
- = 1 – нормальный.

#### 2.1.7.2 Режим «Выключен»

В этом режиме выход AI.x не работает — уровни на физическом канале ввода не воспринимаются. Значение регистра «AI.x нормальный: Значение» автоматически устанавливается в 0.0.

#### 2.1.7.3 Режим «Нормальный»

В этом режиме вход AI.x работает как обычный аналоговый: уровни на физическом канале ввода воспринимаются, оцифровываются средствами АЦП микроконтроллера ПЛК и масштабируются в значение диапазона 0 ... 10 В. Значение уровня доступно в регистре «AI.x нормальный: Значение». Чтобы не нагружать вычислительные мощности микроконтроллера ПЛК, опрос АЦП (в циклическом режиме) выполняется с помощью встроенного аппаратного контроллера DMA (контроллер прямого доступа к памяти). Период выборки данных из канала АЦП равен 50 мс.

Регистр «AI.x нормальный: Значение» - значение уровня:

- = 0.0 ... 10.0 В.

#### 2.1.8 Безопасное состояние каналов вывода

Для каналов вывода доступен режим автоматического перевода их в безопасное состояние.



Для чего необходим этот режим:

- защита от некорректной работы управляющей программы (зависание, зацикливание), если она управляет каналами вывода;
- защита от сбоев в работе сети MODBUS RTU (обрыв кабеля, зависание Master-устройства), если управление каналами вывода осуществляется дистанционно через интерфейс COM.2 ПЛК.

Активация режима:

1) для каналов вывода задать значение регистра «Безопасное состояние, значение»;

2) задать ненулевое значение регистра «Таймер безопасного состояния каналов вывода, время уставки (сек)» (см. подраздел 2.1.10 «Системные команды и настройки»).

Деактивация режима:

1) задать нулевое значение регистра «Таймер безопасного состояния каналов вывода, время уставки (сек)».

Алгоритм работы (при активном режиме):

- таймер безопасного состояния каналов вывода считает (каждую секунду);
- при достижении счетчика таймера значения уставки:
  - таймер останавливается (без сброса счетчика),
  - на каналы вывода выдаются установленные безопасные состояния,
  - зажигается светодиод «ERR» на лицевой панели ПЛК.

Безопасное состояние применяется только для активных каналов вывода (чей режим отличен от режима «Выключен»). При активном режиме необходимо сбрасывать таймер подачей команды через регистр «Таймер безопасного состояния каналов вывода, команда сброса» (см. подраздел 2.1.10 «Системные команды и настройки»). Сброс нужно успеть сделать до завершения счета таймера.

Способ подачи команды сброса:

- через адресуемую переменную управляющей программы,
- через сетевой регистр ModBus (дистанционно через COM.2 ПЛК).

### 2.1.9 Сторожевой таймер

В микроконтроллере ПЛК имеется сторожевой таймер, который позволяет выполнить автоматическую перезагрузку устройства.

Для чего это необходимо:

- защита от сбоя в работе системы исполнения ПЛК (зависание, закливание).

Активация сторожевого таймера:

1) задать ненулевое значение регистра «Сторожевой таймер, время уставки (сек)» (см. подраздел 2.1.10 «Системные команды и настройки»).

Деактивация режима:

1) задать нулевое значение регистра «Сторожевой таймер, время уставки (сек)».

Алгоритм работы (при активном сторожевом таймере):

- сторожевой таймер считает (каждую миллисекунду);
- при достижении счетчика таймера значения уставки:
  - срабатывает автоматический перезапуск микроконтроллера ПЛК.

При активном режиме необходимо сбрасывать таймер подачей команды через регистр «Сторожевой таймер, команда сброса» (см. подраздел 2.1.10 «Системные команды и настройки»). Сброс нужно успеть сделать до завершения счета таймера.

Способы подачи команды сброса:

- через адресуемую переменную управляющей программы,
- через сетевой регистр MODBUS (дистанционно через COM.2 ПЛК).

### 2.1.10 Системная информация

ПЛК, помимо данных по каналам В/В, должен предоставлять, так называемые, данные «о себе» (системная информация):

- закодированное собственное имя и вариант аппаратного исполнения,
- номер версии и дату сборки системы исполнения,
- состояние системы исполнения,
- температура ЦПУ.

Регистр «Кодовое название ПЛК»: 411.

Регистр «Код варианта аппаратного исполнения ПЛК»: 1.

Регистр «Номер версии системы исполнения ПЛК» - это закодированное значение (раскладка в таблице 2.4): 1 (соответствует значению 1.0.0).

Таблица 2.4 - Раскладка номера версии системы исполнения ПЛК

значение регистра (число, беззнаковое целое, 16 бит)															
1															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1				0				0				0			
version::major				version::minor				version::patch				резерв			

Регистр «Дата выпуска системы исполнения ПЛК» - это закодированное значение (раскладка в таблице 2.5): 169 (соответствует значению 09.05).

Регистр «Год выпуска системы исполняя ПЛК»: 2023.

Таблица 2.5 - Раскладка даты выпуска системы исполнения ПЛК

значение регистра (число, беззнаковое целое, 16 бит)															
169															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0
9					5					0					
день					месяц					резерв					

Регистр «Состояние системы исполнения» - это закодированное значение (раскладка в таблице 2.6): 3 (пример).



= TRUE – есть команда.

Регистр «Таймер безопасного состояния каналов вывода, время (сек)»

= 0 — таймер выключен,

= 1 ... 65000 — таймер включен.

Регистр «Сторожевой таймер микроконтроллера ПЛК, команда сброса»

= FALSE – нет команды,

= TRUE – есть команда.

Регистр «Сторожевой таймер микроконтроллера ПЛК, время (мсек)»

= 0 — таймер выключен,

= 1 ... 65000 — таймер включен.

### 2.1.12 Пользовательские регистры данных

ПЛК предоставляет память для хранения данных пользователя:

- 32 регистра для хранения логических значений (типа BOOL),
- 64 регистра для хранения числовых значений.

### 2.1.13 Интерфейс «COM.1»

COM.1 - это последовательный интерфейс RS-485, аппаратно реализованный на базе микросхемы приемо-передатчика MaxLinear SP485EEN-L, которая подключена к интерфейсу UART1 микроконтроллера ПЛК.

RS-485 является полудуплексным интерфейсом (одна линия передачи данных, позволяющая в один момент времени выполняется либо отправку, либо получение данных), а интерфейс UART – полнодуплексный (две независимых линии передачи данных — Rx на прием и Tx на отправку). Микросхема приемо-передатчика RS-485 со стороны UART, помимо входов Rx и Tx, имеет еще один вход «RE» (Receive Enable), с помощью которого выполняется управление направлением передачи данных от UART:

- если низкий уровень (логический «0») на входе «RE», то UART.Tx,

- если высокий уровень (логическая «1») на входе «RE», то UART.Rx.

Управляет входом «RE» микроконтроллер ПЛК - выход «РА11».

Чтобы не нагружать вычислительные мощности микроконтроллера ПЛК, данные через интерфейс UART1 передаются с помощью встроенного аппаратного контроллера DMA (контроллер прямого доступа к памяти).

В зависимости от режима работы ПЛК, интерфейс COM.1 выполняет следующие функции: обновление встроенного ПО, отладка управляющей программы в режиме реального времени.

Если ПЛК работает в режиме «Обновление встроенного ПО» (BOOT):

- 1) От программы-программатора на вход интерфейса COM.1 поступают данные (машинный код системы исполнения или управляющей программы);
- 2) Полученные данные передаются системному загрузчику микроконтроллера ПЛК; [33]
- 3) Системный загрузчик записывает данные в соответствующую область памяти программ микроконтроллера ПЛК; [33]
- 4) Системный загрузчик передает информацию о результате записи данных в память программ на выход интерфейса COM.1 и переходит в режим ожидания поступления новых данных. [33]

Если ПЛК работает в режиме «Работа» (RUN):

- 1) От модуля-отладчика среды Veremiz на вход интерфейса COM.1 поступают данные (запросы протокола отладки); [21][26]
- 2) Полученные данные передаются системе исполнения — задаче, реализующей протокол отладки;
- 3) Задача-отладчик системы исполнения обрабатывает запрос и формирует необходимый ответ;
- 4) Сформированный задачей-отладчиком ответ система исполнения передает на выход интерфейса COM.1 и переходит в режим ожидания поступления новых данных.

Сетевые настройки интерфейса COM.1, соответствующие таблице характеристик ПЛК, жестко прописаны в системе исполнения и изменению не подлежат. Работа сетевого интерфейса COM.1 (прием/передача данных) индицируется путем мигания светодиода «NET».

#### 2.1.14 Интерфейс «COM.2»

COM.2 - это последовательный интерфейс RS-485, аппаратно реализованный, также как и COM.1, на базе микросхемы приемо-передатчика MaxLinear SP485EEN-L, но которая подключена к интерфейсу UART2 микроконтроллера ПЛК. Микроконтроллер ПЛК (через выход «РА4») управляет потоком данных через микросхему приемо-передатчика (вход «RE»).

Чтобы не нагружать вычислительные мощности микроконтроллера ПЛК, данные через интерфейс UART2 передаются с помощью встроенного аппаратного контроллера DMA (контроллер прямого доступа к памяти). Интерфейс COM.2, вне зависимости от режима работы ПЛК, выполняет только одну функцию — передача значений регистров ПЛК по протоколу MODBUS RTU (режим Slave).

MODBUS – это открытый и простой в реализации коммуникационный протокол, основанный на архитектуре «Ведущий (Master) — Ведомый (Slave)». Широко применяется для организации связи между различными электронными устройствами. Может использоваться для передачи данных через последовательные линии связи (RS-232, RS-422, RS-485 – MODBUS RTU) и сети TCP/IP (MODBUS TCP). [30]

Master-устройство является главным в сети MODBUS:

- 1) формирует и отправляет запросы для Slave-устройства,
- 2) получает ответы на запрос от Slave-устройства.

Slave-устройство является ведомым (второстепенным) в сети MODBUS:

- 1) ожидает запросов от Master-устройства,
- 2) формирует и отправляет ответ на запрос.

В сети MODBUS может быть только одно Master-устройство и несколько Slave-устройств (в зависимости от топологии сети: например, до 32-х устройств в сети RS-485). [30]

MODBUS, помимо протокола передачи данных, описывает способ организации хранилища данных — данные группируются по таблицам в зависимости от их типа и способа доступа к ним (чтение / запись). Описание структуры таблиц MODBUS приводится в подразделе 2.3 «Карта регистров данных». Следует отметить, что, согласно стандарту, «полезными» данными в MODBUS является понятие - «регистр» (к «неполезным» данным относятся различные заголовки протокола MODBUS). Регистр — это число вида «16-битное беззнаковое целое» (иное название - СЛОВО, WORD или в машинной 32-битной интерпретации — ПОЛУСЛОВО, HALFWORD).

MODBUS определяет следующий порядок следования байт регистров:

- при передаче одного регистра:
  - 1-0 (старшим байтом вперед)
- при передаче нескольких регистров (например, многобайтных чисел):
  - 1-0 3-2 ... (младшим регистром вперед и старшим байтом вперед)

Например, необходимо передать регистр со значением типа REAL (32-битное число, разложенное на два 16-битных регистра) — порядок передачи байт будет следующий:

- 1) Старший байт Первого регистра (1)
- 2) Младший байт Первого регистра (0)
- 3) Старший байт Второго регистра (3)
- 4) Младший байт Второго регистра (2)

Сетевые настройки интерфейса COM.2, соответствующие таблице характеристик ПЛК, жестко прописаны в системе исполнения и изменению не подлежат. Работа сетевого интерфейса COM.2 (прием/передача данных) индицируется путем мигания светодиода «NET».



## 2.2 Схема многозадачности системы исполнения

Система исполнения «ПЛК411» включает в себя:

- операционную систему реального времени (ОСРВ),
- комплекс программ (процессы, задачи, обработчики событий):
  - для обеспечения работы каналов В/В в заданном режиме,
  - для обмена данными через сетевые интерфейсы,
  - для выполнения управляющей программы.

Схема многозадачности системы приведена в ПРИЛОЖЕНИИ В.

### 2.2.1 Процесс «DI»

Процесс обеспечивает работу каналов дискретного ввода и включает задачи: «CPU.EXTI», «DI\_IRQ\_TASK», «DI\_SET\_TASK».

#### 2.2.1.1 Задача «CPU.EXTI»

Это задача-обработчик аппаратного прерывания физического канала дискретного ввода (при смене фронта сигнала), работающая в блокирующем режиме.

Рабочий цикл задачи:

- 1) Ожидание прерывания от физического канала;
- 2) Чтение уровня сигнала с физического канала, вызывавшего прерывание;
- 3) Исполнение алгоритма «фильтр-антидребезг» (задержка по времени);
- 4) Оправка значения канала ввода в задачу «DI\_IRQ\_TASK» через очередь «DI\_IRQ\_Q»;
- 5) Переход к п.(1).

#### 2.2.1.2 Задача «DI\_IRQ\_TASK»

Это задача ОСРВ, работающая в блокирующем режиме.

Рабочий цикл задачи:

- 1) Ожидание данных в очереди «DI\_IRQ\_Q» (блокирующий режим);
- 2) Чтение данных из очереди «DI\_IRQ\_Q»;
- 3) Ожидание мьютекса «DI\_SET\_MX» (блокирующий режим);
- 4) Чтение настроек канала из общей памяти «настройки»;
- 5) Выполнение алгоритма, соответствующего режиму работы канала;
- 6) Ожидание мьютекса «DI\_DATA\_MX» (блокирующий режим);
- 7) Сохранение данных в общей памяти «данные»;
- 8) Отправка данных в задачу «DATA\_TASK» через очередь «DI\_DATA\_Q»;
- 9) Переход к п.(1).

### 2.2.1.3 Задача «DI\_SET\_TASK»

Это задача OCPB, работающая в блокирующем режиме.

Рабочий цикл задачи:

- 1) Ожидание данных в очереди «DI\_SET\_Q» (блокирующий режим);
- 2) Чтение данных из очереди «DI\_SET\_Q»;
- 3) Обработка настроек / команд для канала;
- 4) Ожидание мьютекса «DI\_SET\_MX» (блокирующий режим);
- 5) Сохранение настроек канала в общей памяти «настройки»;
- 6) Ожидание мьютекса «DI\_DATA\_MX» (блокирующий режим);
- 7) Сброс значений счетчиков (если была команда) в общей памяти «данные»;
- 8) Отправка результата обработки настроек / команд, а также обновленных данных (если был сброс счетчиков) в задачу «DATA\_TASK» через очередь «DI\_DATA\_Q»;
- 9) Переход к п.(1).

### 2.2.2 Процесс «DO»

Процесс обеспечивает работу каналов дискретного вывода и включает задачи: «CPU.TIM», «DO\_SET\_TASK».

#### 2.2.2.1 Задача «CPU.TIM»

Это задача аппаратного таймера микроконтроллера ПЛК, работающая в блокирующем режиме. Таймер всегда настроен на аппаратный режим «ШИМ» (вне зависимости от режима работы канала) и с помощью этой задачи управляет физическим каналом дискретного вывода (для каждого канала свой таймер). Настройка таймера (также его включение / выключение) выполняется через аппаратные регистры из задачи «DO\_SET\_TASK».

Рабочий цикл задачи:

- 1) Ожидание прерывания переполнения счетчика таймера;
- 2) Чтение аппаратных регистров с настройками;
- 3) Выполнение алгоритма аппаратного режима «ШИМ»;
- 4) Установка уровня сигнала на физический канал вывода;
- 5) Переход к п.(1).

#### 2.2.2.2 Задача «DO\_SET\_TASK»

Это задача ОСПВ, работающая в блокирующем режиме.

Рабочий цикл задачи:

- 1) Ожидание данных в очереди «DO\_SET\_Q» (блокирующий режим);
- 1) Чтение данных из очереди «DO\_SET\_Q»;
- 2) Обработка настроек / команд для канала;
- 3) Сохранение настроек канала в общей памяти «настройки»;
- 4) Установка аппаратных регистров таймера для задачи «CPU.TIM»;
- 5) Отправка результата обработки настроек / команд в задачу «DATA\_TASK» через очередь «DO\_DATA\_Q»;
- 6) Переход к п.(1).

#### 2.2.3 Процесс «AI»

Процесс обеспечивает работу каналов аналогового ввода и включает задачи: «CPU.ADC», «AI\_TIM», «AI\_SET\_TASK».

### 2.2.3.1 Задача «CPU.ADC»

Это задача аппаратного АЦП и контроллера прямого доступа к памяти (DMA) микроконтроллера ПЛК, работающая в блокирующем режиме. АЦП настроен и работает в связке с DMA. Настройка АЦП (также его включение / выключение) выполняется через аппаратные регистры из задачи «AI\_SET\_TASK».

Рабочий цикл задачи:

- 1) Ожидание команды «Запуск конвертации»;
- 2) Чтение аппаратных регистров с настройками;
- 3) АЦП выбирает начальный канал ввода;
- 4) Опрос физических каналов аналогового ввода (включая встроенный в микроконтроллер датчик температуры):
  - а) АЦП измеряет (оцифровка) уровень сигнала на канале ввода,
  - б) АЦП передает управление DMA,
  - в) DMA копирует значение измерения в общую память «измерения»,
  - г) DMA передает управление АЦП,
  - д) АЦП увеличивает внутренний счетчик измерений для канала,
  - е) если счетчик измерений АЦП меньше уставки, то переход к п.(а),
  - ж) АЦП сбрасывает счетчик измерений,
  - з) АЦП переключается на следующий канал ввода,
  - и) если было переключение на следующий канал, то переход к п.(а);
- 5) АЦП вызывает аппаратное прерывание «Конец конвертации»:
  - а) вызывает функцию-обработчик, запускающего таймер «AI\_TIM».
- 6) Переход к п.(1).

Уставка счетчика измерений находится в настройках системы исполнения (исходный код процесса «AI»): константа PLC\_AI\_ADC\_CHANNEL\_MEASURES (по-умолчанию, 25 измерений на один канал). Измерение — это код АЦП.

### 2.2.3.2 Задача «AI\_TIM\_TASK»

Это задача программного таймера «AI\_TIM», работающего в блокирующем режиме. С помощью этого программного таймера реализуется периодичность опроса ( $T=100\text{мсек}$ ) каналов аналогового ввода (включая датчик температуры, встроенный в микроконтроллер ПЛК). Программный таймер работает в режиме «Одно выполнение»: запускается задачей «AI\_SET\_TASK» при смене режима канала AI с «выключено» на «опрос» или сигналом от функции-обработчика аппаратного прерывания АЦП «Конец конвертации». При достижении счетчиком таймера значения периодичности опроса, таймер останавливается и генерирует программное прерывание, которое запускает задачу «AI\_TIM\_TASK».

Рабочий цикл задачи:

- 1) Ожидание прерывания от программного таймера «AI\_TIM»;
- 2) Чтение измерений АЦП (поканальное);
- 3) Усреднение измерений (поканальное);
- 4) Преобразование усредненных данных (поканальное масштабирование кода АЦП в значение напряжения от 0 до 10 В);
- 5) Сохранение преобразованных данных в общую память «данные»;
- 6) Отправка преобразованных данных в задачу «DATA\_TASK» через очередь «AI\_DATA\_Q»;
- 7) Отправка команды «Запуск конвертации» для АЦП;
- 8) Переход к п.(1).

### 2.2.3.3 Задача «AI\_SET\_TASK»

Это задача ОСРВ, работающая в блокирующем режиме.

Рабочий цикл задачи:

- 1) Ожидание данных в очереди «AI\_SET\_Q» (блокирующий режим);
- 2) Чтение данных из очереди «AI\_SET\_Q»;
- 3) Обработка настроек / команд для канала;
- 4) Ожидание мьютекса «AI\_SET\_MX» (блокирующий режим);

- 5) Сохранение настроек канала в общей памяти «настройки»;
- 6) Отправка результата обработки настроек / команд в задачу «DATA\_TASK» через очередь «AI\_DATA\_Q»;
- 7) Переход к п.(1).

## 2.2.4 Процесс «MODBUS»

Процесс обеспечивает передачу данных через интерфейс COM.2 ПЛК и реализацию протокола ModBus RTU и включает задачи: «MODBUS\_TASK».

### 2.2.4.1 Задача «MODBUS\_TASK»

Это задача ОСРВ, работающая в блокирующем режиме. Прием/передача данных через интерфейс COM.2 выполняется с помощью контроллера прямого доступа к памяти (DMA).

Рабочий цикл аппаратного приема запроса:

- 1) COM.2.Rx ожидает приема запроса;
- 2) Прием запроса:
  - а) COM.2.Rx передает управление DMA;
  - б) DMA копирует данные из COM.2.Rx в общую память «данные»,
  - в) DMA передает управление COM.2.Rx,
  - г) если есть пауза (2,5 ... 3,5 мсек) в поступлении, то переход к п.(3), иначе — переход к п.(а);
- 3) COM.2.Rx вызывает аппаратное прерывание, в обработчике которого отправляется команда «Конец приема данных» в задачу «MODBUS\_TASK» через очередь «COM2\_Q»;
- 4) Переход к п.(1).

Рабочий цикл задачи «MODBUS\_TASK»:

- 1) Ожидание данных в очереди «COM2\_Q» (блокирующий режим);
- 2) Чтение данных из очереди «COM2\_Q»;
- 3) Обработка команд:
  - а) если команда «Конец приема данных», то переход к п.(5), иначе п.(б),

- б) если команда «Конец отправки данных», то переход к п.(4), иначе п.(а);
- 4) Передача управления COM.2.Rx и переход к п.(7);
- 5) Обработка запроса:
  - а) чтение запроса из общей памяти «данные»,
  - б) ожидание мьютекса «REG\_MX»,
  - в) запись данных в таблицы регистров «MODBUS TABLES»,
  - г) формирование ответа,
  - д) запись ответа в общую память «данные»;
- 6) Отправка команды «Запуск отправки данных» для COM.2.Tx.
- 7) Переход к п.(1).

Рабочий цикл аппаратной отправки ответа:

- 1) COM.2.Tx ожидает команды запуска отправки ответа;
- 2) Отправка ответа:
  - а) COM.2.Tx передает управление DMA;
  - б) DMA копирует данные из общей памяти «данные» в COM.2.Tx,
  - в) DMA передает управление COM.2.Tx,
  - г) если переданы все данные, то переход к п.(3), иначе — п.(а);
- 3) COM.2.Tx вызывает аппаратное прерывание, в обработчике которого отправляется команда «Конец передачи данных» в задачу «MODBUS\_TASK» через очередь «COM2\_Q»;
- 4) Переход к п.(1).

## 2.2.5 Процесс «DATA»

Процесс обеспечивает безопасный обмен данными между задачами и общей памятью регистров данных ПЛК и включает задачи: «DATA\_TASK».

### 2.2.5.1 Задача «DATA\_TASK»

Эта задача ОСРВ, работающая постоянно - циклически проверяя очереди данных от различных процессов.

Рабочий цикл задачи:

- 1) Проверка очереди «DI\_DATA\_Q» (неблокирующий режим):
  - а) если есть данные, то вызов функции «Менеджер данных каналов DI»;
- 2) Проверка очереди «DO\_DATA\_Q» (неблокирующий режим):
  - а) если есть данные, то вызов функции «Менеджер данных каналов DO»;
- 3) Проверка очереди «AI\_DATA\_Q» (неблокирующий режим):
  - а) если есть данные, то вызов функции «Менеджер данных каналов AI»;
- 4) Проверка очереди «REG\_IRQ\_Q» (неблокирующий режим):
  - а) если есть данные, то вызов функции «Менеджер регистров»;
- 5) Переход к п.(1).

Менеджер данных каналов DI:

- 1) Чтение данных из очереди «DI\_DATA\_Q»;
- 2) Ожидание мьютекса «REG\_MX»;
- 3) Запись данных в таблицы регистров «MODBUS TABLES».

Менеджер данных каналов DO:

- 1) Чтение данных из очереди «DO\_DATA\_Q»;
- 2) Ожидание мьютекса «REG\_MX»;
- 3) Запись данных в таблицы регистров «MODBUS TABLES».

Менеджер данных каналов AI:

- 1) Чтение данных из очереди «AI\_DATA\_Q»;
- 2) Ожидание мьютекса «REG\_MX»;
- 3) Запись данных в таблицы регистров «MODBUS TABLES».

Менеджер регистров:

- 1) Чтение данных из очереди «REG\_Q»;
- 2) Проверка идентификатора процесса - назначения данных:
  - «DI»: данные в задачу «DI\_SET\_TASK» через очередь «DI\_SET\_Q»,
  - «DO»: данные в задачу «DO\_SET\_TASK» через очередь «DO\_SET\_Q»,
  - «AI»: данные в задачу «AI\_SET\_TASK» через очередь «AI\_SET\_Q».



## 2.2.6 Процесс «REGISTER CONTROLLER»

Процесс обслуживает таблицы регистров данных «MODBUS TABLES» - выполняет операции: чтение, запись и контроль ключевых регистров. Ключевые регистры заданы в виде списка (прописан в коде системы исполнения). При изменении значения регистра в таблице, его значение (с указанием идентификатора назначения данных) передается в задачу «DATA\_TASK» (менеджер регистров) через очередь «REG\_Q». Процесс не имеет задач ОСРВ и реализуется в виде функций работы с таблицами регистров данных.

## 2.2.7 Процесс «APP»

Процесс обеспечивает исполнение управляющей программы в соответствии с рабочим циклом ПЛК и включает задачи: «APP\_TASK» и «APP\_TIM\_TASK».

### 2.2.7.1 Задача «APP\_TASK»

Это задача ОСРВ, работающая постоянно — циклически выполняя протокол отладки (при наличии запросов от среды Veremiz) и выполняя управляющую программу (по программному таймеру, который организует ее рабочий цикл).

Рабочий цикл задачи:

- 1) Проверка мьютекса «APPD\_MX» (неблокирующий режим):
  - а) если мьютекс свободен, то выполнение алгоритма «Отладка»;
- 2) Проверка семафора «APP\_SEMA» (неблокирующий режим):
  - а) если семафор освобожден, то:
    - захват семафора,
    - чтение регистров каналов ввода,
    - выполнение алгоритма управляющей программы,
    - запись регистров каналов вывода,
    - запуск таймера «APP\_TIM»;
- 3) Переход к п.(1).

#### 2.2.7.2 Задача «APP\_TIM\_TASK»

Это задача программного таймера «APP\_TIM», работающего в блокирующем режиме. С помощью этого программного таймера реализуется периодичность выполнения управляющей программы. Периодичность задается в настройках управляющей программы при ее разработке в среде Veremiz. Программный таймер работает в режиме «Одно выполнение»: запускается задачей «APP\_TASK» при завершении выполнения управляющей программы. При достижении счетчиком таймера значения заданной периодичности, таймер останавливается и генерирует программное прерывание, которое вызывает задачу «APP\_TIM\_TASK».

Рабочий цикл задачи:

- 1) Ожидание прерывания от программного таймера «APP\_TIM»;
- 2) Освобождение семафора «APP\_SEMA»;
- 3) Переход к п.(1).

### 2.3 Карта регистров данных

Доступ ко всем регистрам данных ПЛК предоставляется:

- по сети - через интерфейс COM.2 (протокол MODBUS),
- локально - через адресуемые переменные управляющей программы (протокол Veremiz YAPLC).

Протокол MODBUS подразумевает распределение данных по таблицам в зависимости от типа данных и способа доступа (таблица 2.7). За каждой таблицей закреплен свой код (код функции), за каждым регистром — адрес (в пределах таблицы). Соответственно, в запросе значения регистра (или значений нескольких последовательных регистров) необходимо указать код таблицы и начальный адрес регистра (подробно см. Спецификацию на протокол MODBUS [30]).

Таблица 2.7 - Типы таблиц ModBus

Таблица	Код функции		Тип данных (для одного регистра)
	чтение	запись	
DISCRETE COILS (битовые катушки)	1	15	BOOL (1 бит)
DISCRETE INPUTS (битовые входы)	2		
HOLDING REGISTERS (числовые данные)	3	16	WORD (16 бит)
INPUT REGISTERS (числовые входы)	4		

Принцип адресации в среде программирования Beremiz (протокол YAPLC) приведен в подразделе 1.4.4 «Связь переменных с регистрами каналами В/В». Карта адресов регистров данных ПЛК приведена в ПРИЛОЖЕНИИ А.

## 2.4 Выбор средств разработки

Для начала работы разработчику необходимы следующие инструменты:

- набор специальных программных библиотек;
- компилятор исходного кода и компоновщик (линковщик);
- программная среда разработки;
- программатор-отладчик;
- целевая или оценочная/отладочная плата.

### 2.4.1 Встраиваемая операционная система реального времени

Заданием определено, что разрабатываемая для ПЛК система исполнения должна строиться на основе ОСРВ. Для сравнительного анализа взяты три известные ОСРВ для микроконтроллеров: FreeRTOS, KeilRTX и  $\mu$ C/OS. Результат анализа приведен в таблице 2.8.

Таблица 2.8 - Сравнительный анализ ОСРВ

N п.п.	Показатель	FreeRTOS	KeilRTX	μC/OS
1	Поддержка микроконтроллеров (серия STM32F4*)	большое количество (+)	только ARM (+)	большое количество (+)
2	Портируемость на другие аппаратные платформы (желательно)	высокая (+)	нет (-)	хорошая (+)
3	Требования к памяти микроядро + планировщик + 1 задача ОЗУ (< 35 кБ) память программ (< 90 кБ)	632 Б (+) 4 кБ (+)	648 Б (+) 6 кБ (+)	728 Б (+) 6 кБ (+)
4	Тактирование планировщика (от аппаратного таймера микроконтроллера, 1 такт = 1 мсек)	(+)	(+)	(+)
5	Скорость переключения контекста (высокая)	(+)	(+)	(+)
6	Система настроек (понятная, гибкая)	(+)	(+)	(+)
7	Базовый функционал ядра: <ul style="list-style-type: none"> <li>• планировщик</li> <li>• многозадачность</li> <li>• средства межпроцессного взаимодействия</li> <li>• программные таймеры</li> <li>• контроль стека</li> </ul>	есть есть большой набор  есть есть	есть есть урезанный набор  есть есть	есть есть урезанный набор  есть есть
7	<ul style="list-style-type: none"> <li>• приоритеты задач (настраиваемые)</li> <li>• трассировка / отладка</li> </ul>	есть  есть	есть  есть	есть  есть
8	Стоимость, доступность (бесплатная, свободная, общественная версия)	платная, бесплатная (open-source) (+)	только платная лицензия (-)	частично платная лицензия (-)

## Окончание таблицы 2.8

N п.п.	Показатель	FreeRTOS	KeilRTX	μC/OS
9	Техническая поддержка <ul style="list-style-type: none"> <li>• документация</li> <li>• форум поддержки</li> <li>• примеры / шаблоны</li> </ul>	открытая есть много	открытая есть мало	открытая есть мало
10	Разработчик	FreeRTOS Team, Amazon.com	Arm Holdings	Silicon Labs, Micrium Inc
11	Независимые отзывы	отличные	хорошие	хорошие
12	Сертификаты, гарантии	IEC 61508 SIL3	-	-

В результате, принимается решение использовать OCPB FreeRTOS:

- Многозадачная, мультиплатформенная, бесплатная операционная система жесткого реального времени с открытым исходным кодом;
  - Обладает минимальными системными требованиями, хорошо документирована и имеет большое количество примеров и шаблонов, имеет большое количество положительных отзывов;
  - Большая часть кода написана на языке Си, ассемблерные вставки минимального объема применяются лишь там, где невозможно применить Си из-за специфики конкретной аппаратной платформы; [11]
  - Поддерживается большое количество аппаратных платформ (при непосредственном участии крупных производителей электроники): ARM (Cortex-M0, Cortex-M3, Cortex-M4, Cortex-A), RISC-V, Atmel AVR, Xilinx MicroBlaze, Cortus, Texas Instruments MSP, Microchip PIC, Renesas, SuperH, Freescale Coldfire, Fujitsu MB, Altera Nios II, RM4x и прочие; [11]
  - Существуют, так называемые, официально и неофициально поддерживаемые аппаратные платформы — порты; для одного и того же порта могут поддерживаться несколько средств разработки (например, компиляторы GCC, IAR, Keil);
  - Существуют коммерческие версии:

○ SafeRTOS — это ОСПВ, соответствующая уровню функциональной безопасности SIL3, имеющая такую же функциональную модель, что и FreeRTOS, и ориентированная на применение в системах с высокими требованиями к безопасности, например в медицинской и аэрокосмической отраслях;

○ OpenRTOS отличается от FreeRTOS лишь тем, что поставляется под коммерческой лицензией, с гарантией и непосредственной технической поддержкой производителя (поддержка FreeRTOS осуществляется на уровне «вопрос-ответ» через официальный интернет-форум).

## 2.4.2 Стандартные библиотеки периферии микроконтроллера

STMicroelectronics для облегчения труда разработчиков предоставляет бесплатные стандартные библиотеки периферии для своих микроконтроллеров и, в частности, для семейства STM32 — это CMSIS и HAL. Эти библиотеки реализованы на языке Си, используются в большинстве семейств микроконтроллеров STM, бесплатные, открытые (open-source), хорошо документированы и для них имеется много примеров и шаблонов.

Библиотека CMSIS (Cortex Microcontroller Software Interface Standard) - библиотека стандарта программного обеспечения для ядер ARM Cortex-M (Cortex-M0, M3, M4) микроконтроллеров, являющаяся независимым от производителя уровнем аппаратной абстракции и определяет общие интерфейсы инструментов. Уровень абстракции данной библиотеки - низкий - работа ведется на уровне регистров микроконтроллера. Библиотека включает в себя:

- CMSIS-CORE - макроопределения и функции ядра;
- CMSIS-Driver - макроопределения и функции основных драйверов интерфейсов периферии; содержит API для взаимодействия с программным обеспечением верхнего уровня (сетевые интерфейсы, дисковые и блочные устройства);
- CMSIS-DSP - макроопределения и функции для различных типов данных с фиксированной и плавающей точкой;
- CMSIS-RTOS - макроопределения и функции для ОСПВ (FreeRTOS).

Библиотека HAL (Hardware Abstraction Layer) – стандартная программная библиотека для работы с ядром и периферией микроконтроллера на высоком уровне абстракции и, соответственно, большую возможность переносимости кода. Данная библиотека базируется на библиотеке CMSIS (иными словами — «обертка» над CMSIS). Здесь работа ведется на уровне функций, внутри которых реализованы вызовы из конструкций CMSIS. [34]

### 2.4.3 Среда разработки

Для разработки приложений под процессорную архитектуру ARM существует довольно широкий выбор программных средств разработки, которые включают в себя: редактор исходного кода, средства для подключения и настройки инструментов компиляции и отладки. В таблице 2.9 приведен сравнительный анализ наиболее распространенных сред разработки.

Таблица 2.9 - Распространенные среды разработки для ARM архитектур

N п.п.	Среда разработки	Компилятор, компоновщик	Особенности
1	Eclipse IDE	GNU GCC ARM	Бесплатная, свободно распространяемая, с открытым исходным кодом (open-source). Кроссплатформенная (сборки под Windows, Linux и прочие ОС). Многофункциональная (подходит для программирования на различных языках). Официально поддерживается многими производителями микроэлектроники. Без ограничений.
2	CodeBlocks IDE	GNU GCC ARM	Аналогично Eclipse.
3	Keil $\mu$ Vision	Keil C/C++	С закрытым исходным кодом. Платная: без ограничений. Бесплатная: имеется ограничение разрабатываемой программы в 32 кБ.

## Окончание таблицы 2.9

N п.п.	Среда разработки	Компилятор, компоновщик	Особенности
4	IAR Embedded Workbench	IAR C/C++	С закрытым исходным кодом. Платная: без ограничений. Бесплатная: имеется ограничение разрабатываемой программы в 32 кБ или любой размер программы, но с ограничением работы в 30 дней

По результатам анализа, выбрана среда разработки Eclipse: [28]

- свободно распространяемая и открытая (open-source);
- кроссплатформенная (имеются сборки под различные ОС);
- легко настраивается;
- имеется готовая сборка для разработки под ARM:
  - поддерживается большое количество семейств микроконтроллеров,
  - подключен и настроен компилятор GNU GCC ARM C/C++ (в комплекте),
  - подключен отладчик,
  - интегрированы низкоуровневые библиотеки для ARM,
  - при создании нового проекта генерирует готовый рабочий шаблон исходного кода с подключением всех необходимых низкоуровневых библиотек (настраивается),
  - поддерживается (на уровне генератора готовых шаблонов кода) графическим конфигуратором STM32 CubeMx.

### 2.4.4 Графический конфигуратор микроконтроллера

Помимо стандартных библиотек периферии, фирма STMicroelectronics развивает программный продукт STM32CubeMx, который позволяет при помощи достаточно понятного графического интерфейса (рисунок 2.2) произвести настройку любой имеющейся на борту микроконтроллера периферии.



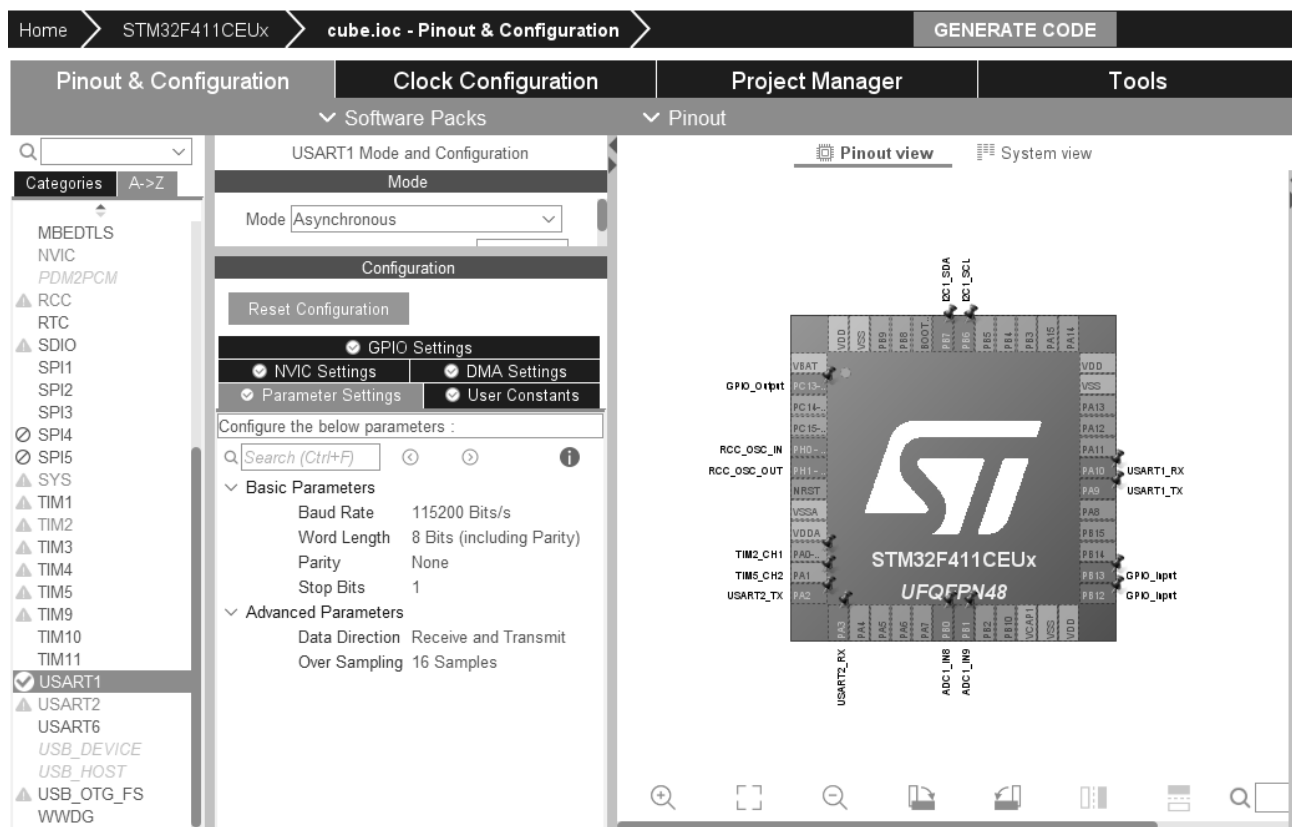


Рисунок 2.2 - Пример окна конфигуратора STM32CubeMx

В процессе работы над проектом пользователь может конфигурировать и настраивать используемое встраиваемое программное обеспечение. Для этого в STM32CubeMx предлагаются: мастер настройки выводов, мастер синхронизации (настройки схемы тактирования), калькулятор энергопотребления, утилита конфигурации периферийных модулей микроконтроллера (GPIO, USART и так далее) и набор библиотек (USB, LwIP - стек TCP/IP, OCPB FreeRTOS и другие). После завершения настройки проекта разработчик генерирует на базе созданной конфигурации программный код на языке C для одной из сред разработки (например, для STM32 MDK-ARM — это среда на базе Eclipse и полностью совместима с ней на уровне конфигурационных файлов).

## 2.4.5 Программатор

Любой микроконтроллер семейства STM32 имеет встроенный загрузчик, позволяющий загружать встраиваемое программное обеспечение, используя, например, интерфейс UART.

Для загрузки требуется:

- 1) преобразователь интерфейсов USB-RS485-UART (рисунок 2.4),
- 2) исполняемый файл встраиваемой программы (.bin, .hex),
- 3) программа-загрузчик.

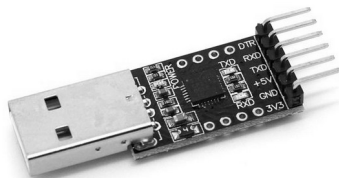


Рисунок 2.3 - Преобразователь интерфейсов USB-RS485-UART

В качестве программы-загрузчика можно использовать:

- Flash Loader Demonstrator: бесплатная и свободно доступная, удобный графически интерфейс, сборка только под ОС Windows;
- stm32flash: бесплатная и свободно доступная, сборки под различные ОС (Windows, Linux), только текстовый интерфейс (консоль). [35]

## 2.5 Разработка системы исполнения

### 2.5.1 Создание нового проекта

В среде Eclipse: [28]

- 1) В меню «Файл» (File) выбрать подменю «Новый» (New);
- 2) В подменю «Новый» выбрать элемент «C/C++ проект» (C/C++ Project);
- 3) В окне «Templates for New C/C++ Project» (рисунок 2.4):
  - а) в левой части выбрать «All»,
  - б) в правой части выбрать «C Managed Build»,
  - в) нажать на кнопку «Next»;

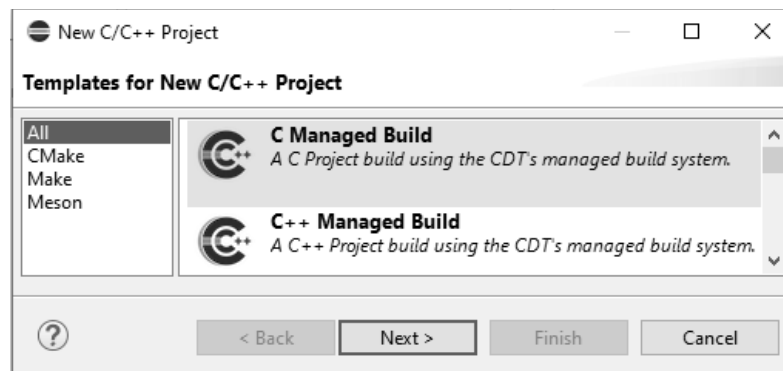


Рисунок 2.4 - Окно «Templates for New C/C++ Project» (выбор шаблона проекта)

4) В окне «C Project» (рисунок 2.5):

- а) ввести имя проекта в поле «Project name»,
- б) выбрать тип проекта «STM32F4xx C/C++ Project»,
- в) выбрать инструмент сборки «ARM Cross GCC»,
- г) нажать на кнопку «Next»;

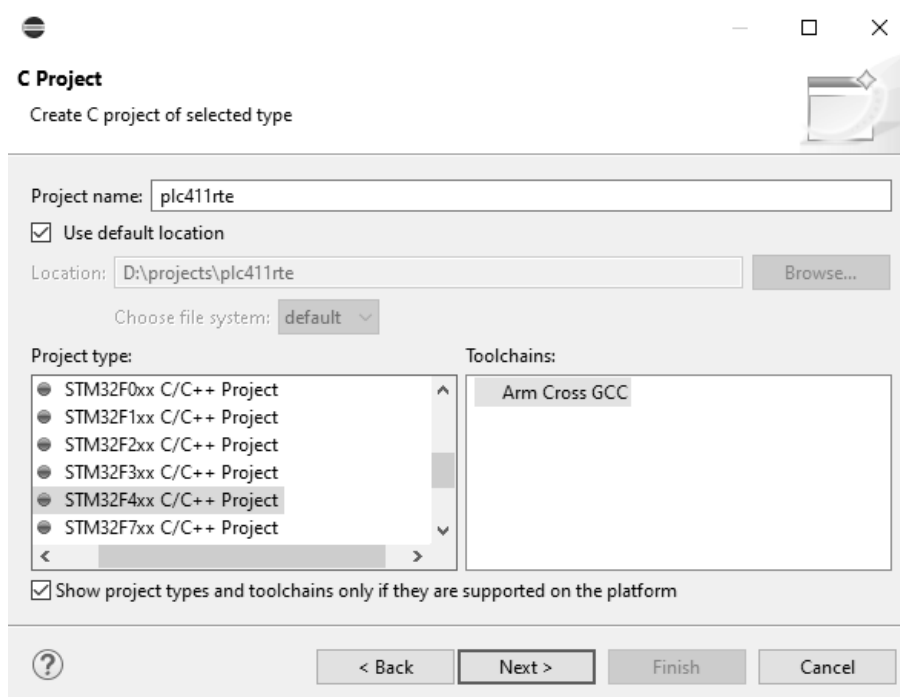


Рисунок 2.5 - Окно «C Project» (выбор типа проекта)

5) В окне «Target processor settings» (рисунок 2.6):

- а) выбрать семейство микроконтроллера «STM32F411xE» («Chip family»),
- б) ввести размер памяти программ «512» («Flash size (kB)»),
- в) ввести частоту внешнего резонатора «25000000» («External clock (Hz)»),

- г) выбрать шаблон кода с примером «Empty» («Content»),
- д) выбрать тип системных вызовов «Freestanding» («Use system calls»),
- е) выбрать тип трассировки «None» («Trace output»),
- ж) остальные опции оставить по-умолчанию:
  - ✓ Check some warnings
  - ✓ Use -Og on debug
  - ✓ Use newlib nano
  - ✓ Exclude unused
- з) нажать на кнопку «Next»;

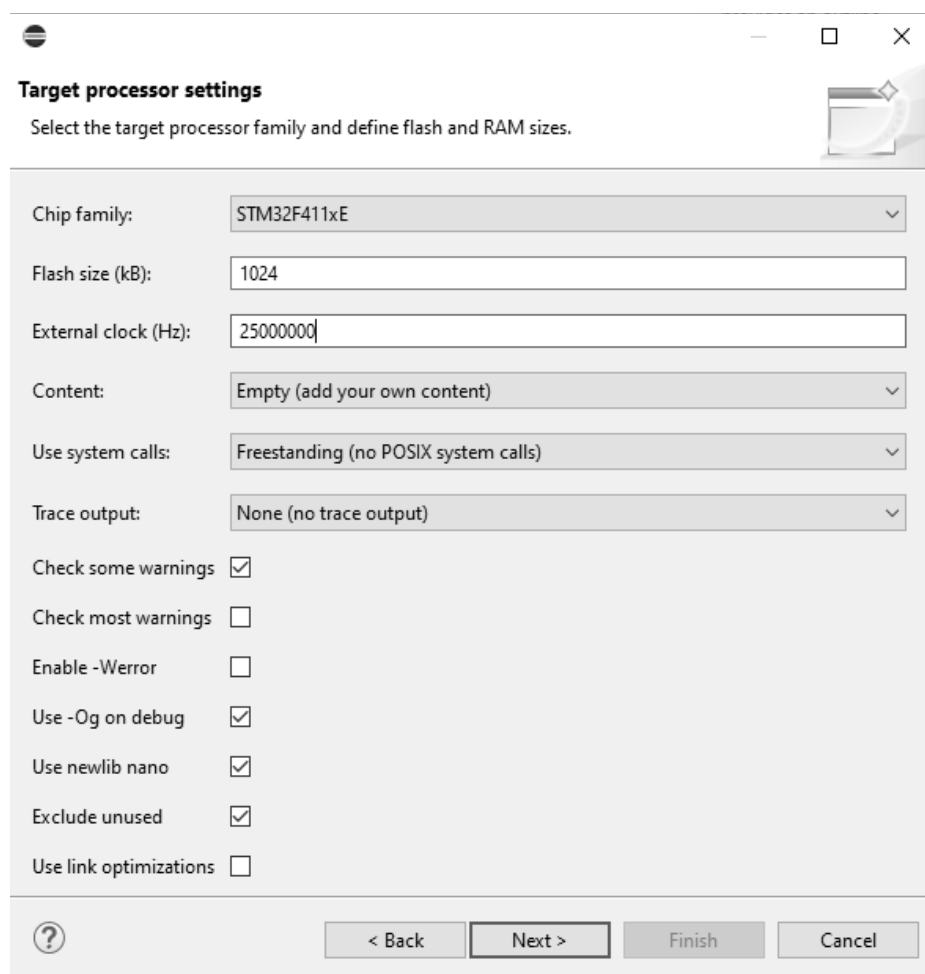


Рисунок 2.6 - Окно «Target processor settings»

- 6) В окне «Folder settings» (рисунок 2.7):
  - а) имена базовых директорий проекта оставить по-умолчанию:
    - include – директория разрабатываемых заголовочных файлов (.h),
    - src – директория разрабатываемых исходных файлов (.c),

- system – директория файлов подключаемых библиотек,
- cmsis – директория (будет вложена в system) библиотеки CMSIS,
- newlib – директория (будет вложена в system) библиотеки newlib,
- ldscripts – директория сценариев компоновщика.

б) нажать на кнопку «Next»;

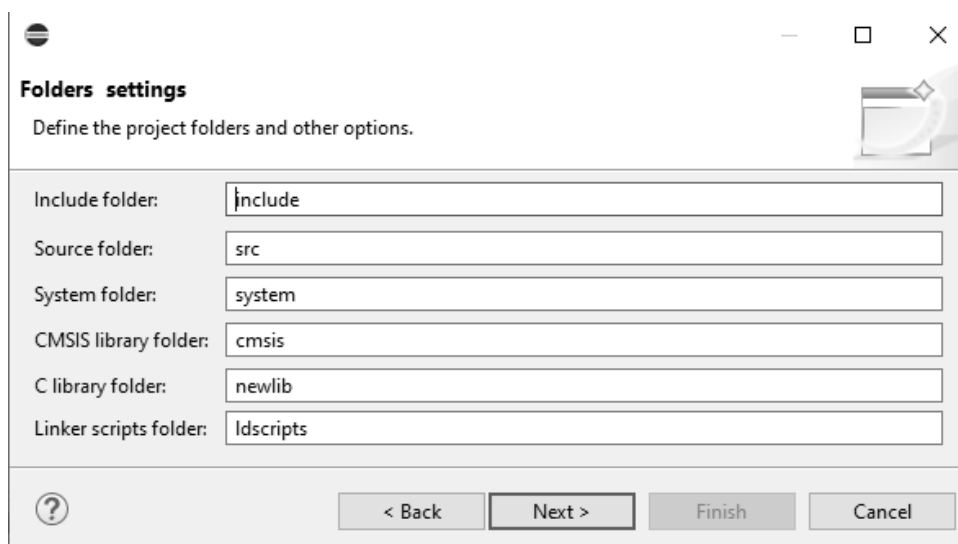


Рисунок 2.7 - Окно «Folder settings»

7) В окне «Select Configurations» (рисунок 2.8):

- а) конфигурации проекта оставить по-умолчанию,
- б) нажать на кнопку «Next»;

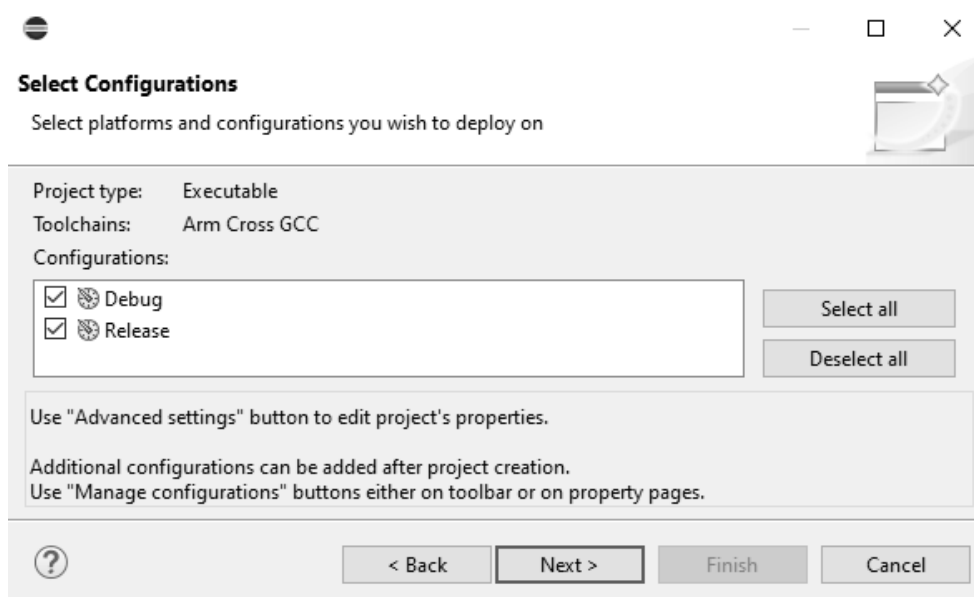


Рисунок 2.8 - Окно «Select Configurations»

8) В окне «GNU Arm Cross Toolchain» (рисунок 2.9):

- а) выбрать имя кросс-компилятора «arm-none-eabi-gcc» («Toolchain name») и указать путь к директории его расположения («Toolchain path»),
- б) нажать на кнопку «Finish».

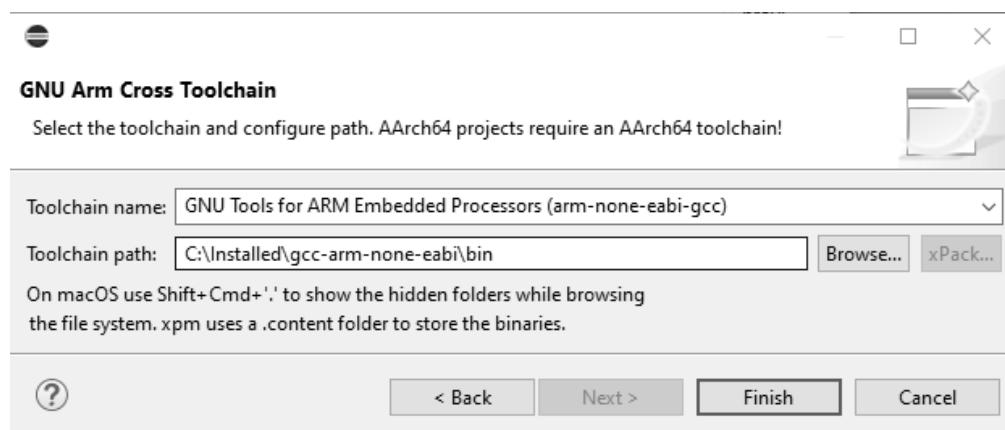


Рисунок 2.9 - Окно «GNU Arm Cross Toolchain»

В результате сформируются файловая система нового проекта, куда будут по-умолчанию интегрированы базовые стандартные системные библиотеки и некоторые шаблонные файлы: ARM (макрофункции работы с регистрами ядра микроконтроллера), CMSIS, DIAG (диагностика), newlib (системные вызовы) [31], HAL [34], скрипты компоновщика (ldscripts), шаблон главного файла main.c

9) Привести файловую структуру нового проекта к следующему виду (назначение директорий см. в таблице 2.10, результат — рисунок 2.10):

- а) в директориях include и src создать поддиректории:
  - stm32f4, sensors, freertos, beremiz;
- б) в директории system создать поддиректории:
  - stm32f4, freertos / include, freertos / src, beremiz / include, beremiz / src, matiec;
- в) файл include/stm32\_assert.h переместить в include/stm32f4/;
- г) файл include/stm32f4xx\_hal\_conf.h переместить в include/stm32f4/;
- д) файл src/initialize-hardware.c переместить (с переименованием) в src/stm32f4/stm32f4xx\_hal\_init.c;
- е) файл src/stm32f4xx\_hal\_msp.c переместить в src/stm32f4/;
- ж) файл src/write.c переместить в src/stm32f4/;

- з) директорию system/include переместить (со всем содержимым) в system/stm32f4/;
- и) директорию system/src переместить (со всем содержимым) в system/stm32f4/;
- к) из директории ldscripts удалить все файлы \*.ld.

Таблица 2.10 - Назначение директорий файловой структуры проекта

Директория	Назначение
Разрабатываемый аппаратнозависимый код	
ldscripts /	скрипты компоновщика (.ld)
include / stm32f4 /	драйверы периферии микроконтроллера
src / stm32f4 /	- заголовочные файлы (.h) - файлы исходного кода (.c)
Разрабатываемый аппаратнонезависимый код	
include / sensors /	обработчики данных сенсоров
src / sensors /	- заголовочные файлы (.h) - файлы исходного кода (.c)
include / freertos /	обработчики процессов ОСРВ
src / freertos /	- заголовочные файлы (.h) - файлы исходного кода (.c)
include / beremiz /	обработчики системных FB / FBD Beremiz
src / beremiz /	- заголовочные файлы (.h) - файлы исходного кода (.c)
include /	прочие настройки и алгоритмы
src /	- заголовочные файлы (.h) - файлы исходного кода (.c)
Подключаемые сторонние системные библиотеки	
system / stm32f4 / include /	ARM Cortex-M, CMSIS, HAL, newlib, diag
system / stm32f4 / src /	- заголовочные файлы (.h) - файлы исходного кода (.c)
system / freertos / include /	FreeRTOS
system / freertos / src /	- заголовочные файлы (.h) - файлы исходного кода (.c)
system / matiec /	компилятор MatIEC
	- заголовочные файлы (.h)

## Окончание таблицы 2.10

Директория	Назначение
Системные модули Beremiz YAPLC (оптимизированные для данной системы исполнения)	
system / beremiz / include / system / beremiz / src /	взаимодействие среды исполнения и управляющей программы, протокол отладки Beremiz YAPLC - заголовочные файлы (.h) - файлы исходного кода (.c)

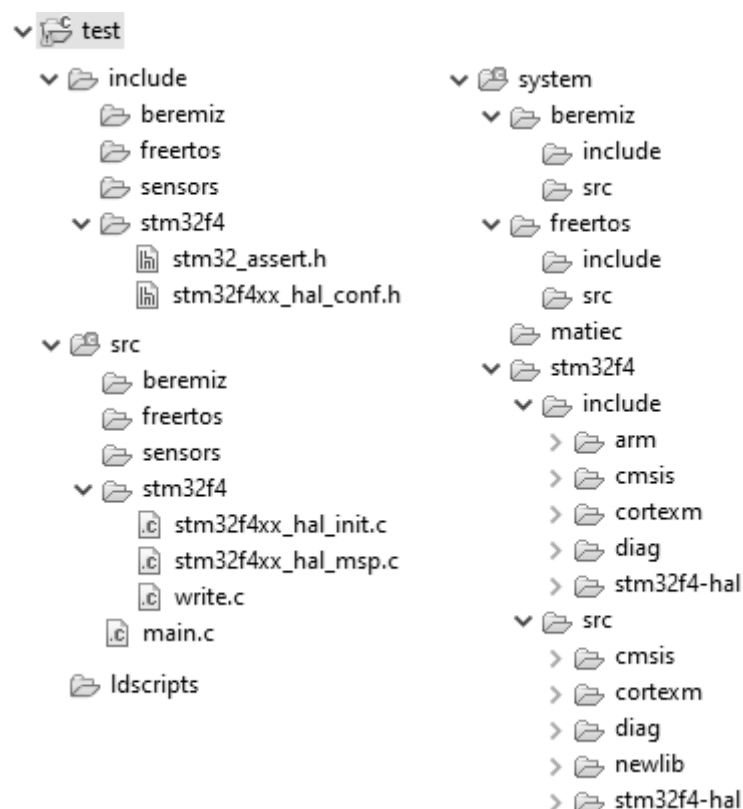


Рисунок 2.10 - Файловая структура нового проекта

### 2.5.2 Подключение FreeRTOS

Дистрибутив FreeRTOS свободно доступен в исходных кодах на языке Си: ядро системы, многочисленные порты для различных аппаратных платформ, примеры. Большая часть файлов не потребуется.

Каждая аппаратная платформа требует небольшой части кода ядра, которая реализует взаимодействие ОСРВ с этой платформой. Весь платформозависимый



код находится в директории source/portable дистрибутива ОСПВ, где он сгруппирован по компиляторам (например, IAR, GCC и т. д.) и, непосредственно, по аппаратным платформам (например, ARM\_CM0, ARM\_CM3, ARM\_CM4F и т. д.). Как правило, платформозависимый код представлен в виде двух файлов: portmacro.h и port.c. [15]

Каждый раз при создании любого объекта (задача, очередь, программный таймер, семафор и так далее) ядро ОСПВ выделяет для этого объекта память из системной кучи — области памяти, доступной для динамического размещения в ней переменных. FreeRTOS поставляется с пятью схемами выделения памяти, которые содержатся в исходных файлах, расположенных в директории source/portable/MemMang. В данной работе будет использоваться схема heap\_1.c:

- реализует самую минимальную по возможности версию функций выделения памяти типа malloc();
- распределение памяти является детерминированным (статическим) — распределяется только один раз при инициализации и во время работы не изменяется;
  - память представляет собой обычный массив постоянного размера;
  - конфигурация определяется в файле FreeRTOSConfig.h;
  - освобождение памяти не предусматривается.

Чтобы подключить FreeRTOS к проекту, необходимо:

- 1) Распаковать архив дистрибутива;
- 2) Скопировать исходный код ядра:
  - а) все файлы FreeRTOS/Source/include/\*.h → system/freertos/include/
  - б) все файлы FreeRTOS/Source/\*.c → system/freertos/src/
- 3) Скопировать исходный код схемы управления памятью:
  - а) файл FreeRTOS/Source/portable/MemMang/heap\_1.c → system/freertos/src/
- 4) Скопировать аппаратнозависимый код (порт для микроконтроллера):
  - а) файл FreeRTOS/Source/portable/GCC/ARM\_CM4F/portmacro.h → system/freertos/include/
  - б) файл FreeRTOS/Source/portable/GCC/ARM\_CM4F/port.c → system/freertos/src/

5) Скопировать конфигурацию ядра из любого демонстрационного проекта:

а) файл FreeRTOS/Demo/CORTEX\_M4F\_STM32F407ZG-SK  
/FreeRTOSConfig.h → include/freertos/

### 2.5.3 Подключение MatIEC

Для реализации некоторых механизмов взаимодействия системы исполнения и управляющей программы, реализованной в среде Beremiz, требуются макроопределения из заголовочных файлов компилятора MatIEC. Эти заголовочные файлы находятся в дистрибутиве Beremiz.

Чтобы подключить заголовочные файлы MatIEC, необходимо:

1) Скопировать все файлы Beremiz/matiec/lib/C/\*.h → system/matiec/

### 2.5.4 Код проекта

#### 2.5.4.1 Конфигурация системы исполнения

В файле config.h (директория include/stm32f4/) определены:

- код ПЛК,
- код варианта исполнения ПЛК,
- версия системы исполнения ПЛК,
- дата сборки системы исполнения ПЛК,
- настройки включения/выключения компиляции модулей,
- настройки вывода отладочной информации,
- значения частот тактирования,
- приоритеты прерываний,
- приоритеты каналов контроллера DMA,
- приоритеты задач ОСРВ,
- настройки интерфейса COM.2.

#### 2.5.4.2 Настройки системы тактирования микроконтроллера

В файле `stm32f4xx_hal_init.c` [34] (директория `src/stm32f4/`) определены функции библиотеки HAL, которые автоматически выполняются после перезагрузки микроконтроллера:

```
void __initialize_hardware(void)
```

Эта функция запускается после перезагрузки ядра Cortex-M микроконтроллера и выполняет: инициализацию компонентов библиотеки HAL, конфигурирование и активацию системы тактирования ядра микроконтроллера.

```
void __attribute__((weak)) SystemClock_Config(void)
```

Эта функция определяет и активирует конфигурацию системы тактирования ядра микроконтроллера.

#### 2.5.4.3 Конфигурация FreeRTOS

В файле `FreeRTOSConfig.h` (директория `include/freertos/`) определены конфигурационные параметры ядра ОСРВ FreeRTOS. Конфигурационных параметров довольно много, поэтому в файле `FreeRTOSConfig.h` определяются только те параметры, значения которых отличаются от значений «по-умолчанию» (т. е., для параметров, не определенных в файле `FreeRTOSConfig.h`, используются значения «по-умолчанию»). Подробное описание каждого параметра можно найти на сайте разработчика FreeRTOS ([freertos.org](http://freertos.org), раздел: Kernel / Developer Docs / `FreeRTOSConfig.h`). [27]

#### 2.5.4.4 Определение элементов ядра FreeRTOS

В файлах `rtos.h` и `rtos.c` (директория `include/freertos/` и `src/freertos/`) определены элементы ядра FreeRTOS:

- очереди:
  - REG\_Q,
  - COM2\_Q,
  - DI\_IRQ\_Q, DI\_SET\_Q, DI\_DATA\_Q,
  - DO\_SET\_Q, DO\_DATA\_Q,

- AI\_SET\_Q, AI\_DATA\_Q;
- мьютексы:
  - REG\_MX,
  - DI\_SET\_MX, DI\_DATA\_MX,
  - AI\_SET\_MX,
  - APPD\_MX;
- семафоры:
  - APP\_SEMA;
- программные таймеры:
  - AI\_TIM,
  - APP\_TIM;
- задачи:
  - MODBUS\_TASK,
  - DATA\_TASK,
  - DI\_IRQ\_TASK, DI\_SET\_TASK,
  - DO\_SET\_TASK,
  - AI\_SET\_TASK,
  - APP\_TASK.

Для очередей определяются:

- {ИМЯ\_ОЧЕРЕДИ} – переменная очереди (неинициализированная);
- {ИМЯ\_ОЧЕРЕДИ}\_ISZ – размер одного элемента очереди в байтах; размер можно задавать с помощью вызова функции sizeof({ТИП ДАННЫХ});
- {ИМЯ\_ОЧЕРЕДИ}\_SZ – максимальное количество элементов в очереди (иначе - размер очереди).

Для мьютекса определяются:

- {ИМЯ\_МЬЮТЕКСА} – переменная мьютекса (неинициализированная).

Для семафора определяются:

- {ИМЯ\_СЕМАФОРА} – переменная семафора (неинициализированная).

Для программного таймера определяются:

- {ИМЯ\_ТАЙМЕРА} – переменная таймера (неинициализированная);
- {ИМЯ\_ТАЙМЕРА}\_NAME – псевдоним таймера (строка);
- {ИМЯ\_ТАЙМЕРА}\_TM – время таймера ( $\geq 1$  мсек).

Для задачи определяются:

- {ИМЯ\_ЗАДАЧИ}\_NAME – псевдоним задачи (строка);
- {ИМЯ\_ТАЙМЕРА}\_STACK\_SZ – размер стека памяти для задачи;
- {ИМЯ\_ТАЙМЕРА}\_PRIORITY – приоритет задачи.

#### 2.5.4.5 Работа с регистрами данных

В файле reg-map.h (директория include/) определены элементы (константы и макрофункции), описывающие карту адресов регистров данных.

В файлах reg.h и reg.c (директория include/) определены элементы (константы, макрофункции, структуры, функции и массивы таблиц), необходимые для создания таблиц регистров данных и работы с ними. Здесь же реализован функционал процесса "REGISTER CONTROLLER" (ограничен макроопределением `#ifdef RTE_MOD_REG_CTRL`).

В файлах reg-init.h и reg-init.c (директория include/) определены функции инициализации таблиц регистров данных. Также здесь определены функции сброса значений регистров в состояние «заводские настройки» (значения «по умолчанию»).

#### 2.5.4.6 Процесс «DI»

Аппаратнозависимая часть процесса «DI» реализована в файлах: di.h и di.c (см. include/stm32f4/ и src/stm32f4/) - где определены элементы (константы, макрофункции, структуры, функции и массивы данных), необходимые для инициализации каналов «DI» и работы с ними на низком уровне (уровне периферии микроконтроллера).

Аппаратнонезависимая часть процесса «DI» реализована в файлах: rto-di.h и rto-di.c (см. include/freertos/ и src/freertos/) - где определены функции-обработчики и функции задач.

#### 2.5.4.7 Процесс «DO»

Аппаратнозависимая часть процесса «DO» реализована в файлах: do.h, do.c, tim2.h, tim2.c, tim5.h, tim5.c, tim.h и tim.c (см. include/stm32f4/ и src/stm32f4/) - где определены элементы (константы, макрофункции, структуры, функции и массивы данных), необходимые для инициализации каналов «DO» и работы с ними на низком уровне (уровне периферии микроконтроллера). Управление каналом «DO.0» во всех режимах осуществляется с помощью аппаратного таймера TIM2 (канал 1) микроконтроллера. Таймер настраивается на работу в режиме «ШИМ», а управление каналом вывода осуществляется путем изменения длительности импульса. Управление каналом «DO.1» во всех режимах осуществляется с помощью аппаратного таймера TIM5 (канал 2) микроконтроллера. Таймер настраивается на работу в режиме «ШИМ», а управление каналом вывода осуществляется путем изменения длительности импульса.

Аппаратнонезависимая часть процесса «DO» реализована в файлах: rto-do.h и rto-do.c (см. include/freertos/ и src/freertos/) - где определены функции-обработчики и функции задач.

#### 2.5.4.8 Процесс «AI»

Аппаратнозависимая часть процесса «AI» реализована в файлах: ai.h и ai.c (см. include/stm32f4/ и src/stm32f4/) - где определены элементы (константы, макрофункции, структуры, функции и массивы данных), необходимые для инициализации каналов «AI» и работы с ними на низком уровне (уровне периферии микроконтроллера).

Аппаратнонезависимая часть процесса «AI» реализована в файлах: rtos-ai.h и rtos-ai.c (см. include/freertos/ и src/freertos/) - где определены функции-обработчики и функции задач.

#### 2.5.4.9 Процесс «MODBUS»

Аппаратнозависимая часть процесса «MODBUS» реализована в файлах: uart2.h, uart2.c, uart.h и uart.c, (см. include/stm32f4/ и src/stm32f4/) - где определены элементы (константы, макрофункции, структуры, функции и массивы данных), необходимые для инициализации каналов интерфейса COM.2 и работы с ним на низком уровне (уровне периферии микроконтроллера).

Аппаратнонезависимая часть процесса «MODBUS» реализована в файлах: rtos-com2.h, rtos-com2.c (см. include/freertos/ и src/freertos/) - где определены функции-обработчики и функции задач. Сам протокол «MODBUS RTU» реализован в файлах: mbrtu.h и mbrtu.c (см. include/ и src/).

#### 2.5.4.10 Процесс «DATA»

Процесс реализуется в исходных файлах аппаратнонезависимой части: rtos-data.h и rtos-data.c.

#### 2.5.4.11 Процесс «APP»

Аппаратнозависимая часть процесса «APP» реализована в файлах: uart1.h, uart1.c, uart.h и uart.c, (см. include/stm32f4/ и src/stm32f4/) - где определены элементы (константы, макрофункции, структуры, функции и массивы данных), необходимые для инициализации каналов интерфейса COM.1 и работы с ним на низком уровне (уровне периферии микроконтроллера).

Аппаратнонезависимая часть процесса «APP», реализующая функционал для взаимодействия системы исполнения и управляющей программы, разработанной в среде программирования Beremiz, находится в файлах: plc\_abi.h, plc\_app.h и plc\_app.c (см. system/beremiz/include/ и system/beremiz/src/).

					27.03.04.2023.203.23.05 ВКР	Лист
Изм.	Лист	№ докум.	Подпись	Дата		107

Аппаратнонезависимая часть процесса «APP», реализующая функционал протокола отладки Beremiz, находится в файлах: plc\_debug.h, plc\_debug.h (см. system/beremiz/include/ и system/beremiz/src/), rtos-com1.h и rtos-com1.c (см. include/freertos/ и src/freertos/).

Аппаратнонезависимая часть процесса «APP», реализующая функционал ОСРВ, находится в файлах: rtos-app.h и rtos-app.c (см. include/freertos/ и src/freertos/).

#### 2.5.4.12 Главная функция «main»

Главная функция реализована в файле main.c (см. src/) и содержит код:

- инициализация адресного пространства регистров данных,
- инициализация контроллера регистров данных,
- инициализация процесса «DI»,
- инициализация процесса «DO»,
- инициализация процесса «AI»,
- инициализация процесса «DATA»,
- инициализация процесса «MODBUS»,
- запуск планировщика ОСРВ,
- вызов функции-обработчика ошибки запуска планировщика ОСРВ.

### 2.5.5 Сценарий компоновщика кода

Компилятор считывает программный код и преобразует его в набор инструкций. Далее в работу вступает компоновщик (linker) — программа, которая распределяет полученный набор инструкций по заданным адресам памяти. Правила распределения инструкций описываются специальным языком и располагается в файле (\*.ld) — скрипт компоновщика.

Минимум информации, необходимый компоновщику:

- расположение и размер памяти,
- размер и расположение стека и кучи,
- указание точки входа (адрес главной функции main),



- расположение частей программы и данных.

Используя карту распределения памяти микроконтроллера (см. п.2.1.2 «Распределение памяти»), необходимо создать два скрипта компоновщика:

- plc411-rte.ld – для сборки системы исполнения,
- plc411-app.ld – для сборки управляющей программы (будет использован в описании системы исполнения для среды Veremiz).

Вначале скрипта указывается карта распределения памяти, где:

- RAM, RAM\_APP, FLASH, FLASH\_APP - разделы памяти
- (xrw) — атрибуты доступа к памяти (r – чтение, w – запись, x – исполнение)
- ORIGIN – начальный адрес раздела памяти
- LENGTH — размер раздела памяти (К – килобайт)

Далее определяются переменные, используемые в описании сценария: размер стека и его начальный адрес в памяти, размер кучи и ее начальный адрес в памяти, указывается точка входа (ENTRY) - адрес главной функции main.

Далее указывается карта размещения данных по разделам памяти. Каждый раздел дополнительно разбивается на секции (.имя\_секции):

- .isr\_vector - таблица векторов прерываний,
- .inits - инициализированные массивы данных определенных размеров,
- .text - машинный код программы,
- .rodata - константы,
- .data - инициализированные глобальные и статические переменные,
- .bss - неинициализированные глобальные и статические переменные.

## 2.5.6 Файловая структура проекта

Таким образом, после создания проекта, подключения библиотек и формирования кода системы исполнения, файловая структура проекта будет иметь вид (рисунок 2.11 — 2.12).

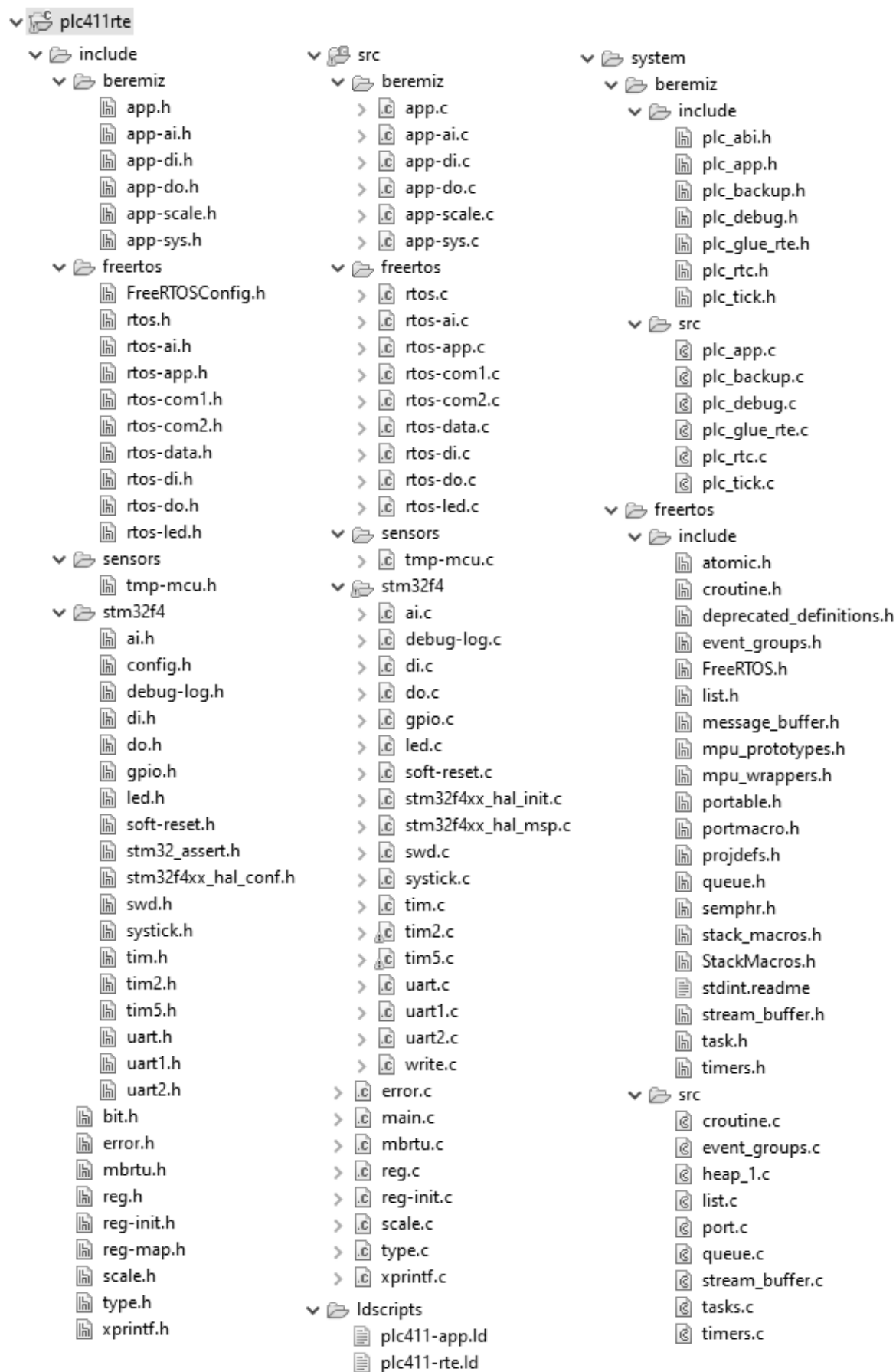


Рисунок 2.11 - Файловая структура проекта (часть 1)

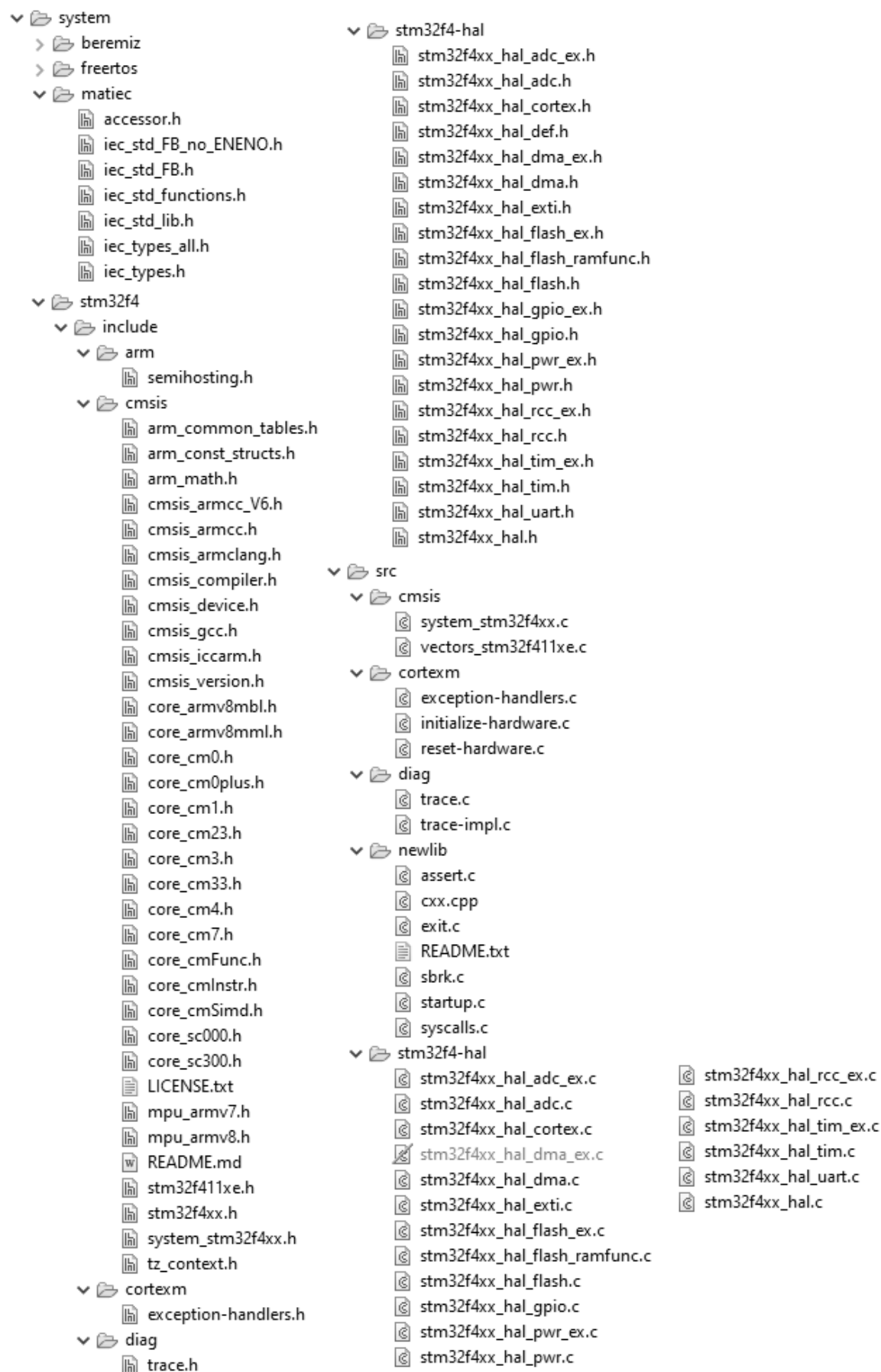


Рисунок 2.12 - Файловая структура проекта (часть 2)

## 2.5.7 Сборка проекта


При сборке проекта автоматически выполняются процедуры:

- 1) компиляция,
- 2) компоновка,
- 3) сохранение результата компоновки в файле (.elf),
- 4) конвертация файла .elf в .hex (и .bin).

.elf (executable and linkable format) - формат исполняемых двоичных файлов. Но, программы-загрузчики, обычно, работают с файлами формата .hex или .bin, которые можно получить путем конвертации файла .elf. Для получения файла .hex в среде Eclipse используется утилита arm-eabi-objcopy (из состава компилятора ARM GCC) – определено в настройках проекта (C/C++ Build / Settings / GNU Arm Cross Create Flash Image / Output file format = Intel HEX).

.hex (Intel Hex) — это шестнадцатеричный формат выходного (объектного) файла. Шестнадцатеричный объектный файл разбит на блоки записей (строки), каждая из которых содержит тип записи, длину, адрес загрузки в память и дополнительную контрольную сумму. Данные в файле кодируются символами ASCII, поэтому данный файл читаем в любом текстовом редакторе. Напрямую загрузить такой файл в память микроконтроллера нельзя, так как он содержит служебную информацию. Поэтому, для загрузки hex-файла используются специальные программные утилиты-загрузчики. Размер hex-файла не соответствует реальному размеру программы.

.bin — это двоичный формат выходного (объектного) файла. Файл нечитаем. Такой формат файла можно напрямую загружать в память микроконтроллера. Размер файла равен размеру памяти, которую займет программа после загрузки в микроконтроллер.

Запуск сборки проекта в среде разработки Eclipse выполняется нажатием на кнопку «Сборка» (). Процесс сборки отображается в консоли, расположенной в нижней части окна среды разработки. Результат сборки — это

					27.03.04.2023.203.23.05 ВКР	Лист
Изм.	Лист	№ докум.	Подпись	Дата		112

завершение сборки с получением выходного (объектного) файла в случае успеха или ошибки с выводом соответствующей информации в консоль.

На рисунке 2.13 показан результат успешной сборки проекта. Статистика по выходному (объектному) файлу приведена в таблице 2.11.

```
Output file is bin\Debug\plc41lrte.elf with size 647.11 KB
Running project post-build steps
arm-none-eabi-size bin\Debug\plc41lrte.elf
arm-none-eabi-objcopy -O ihex bin\Debug\plc41lrte.elf bin\Debug\plc41lrte.elf.hex
  text      data      bss      dec      hex      filename
 58639      136      48904     107679     1a49f     bin\Debug\plc41lrte.elf
arm-none-eabi-objcopy -O binary bin\Debug\plc41lrte.elf bin\Debug\plc41lrte.elf.bin
Process terminated with status 0 (0 minute(s), 2 second(s))
0 error(s), 0 warning(s) (0 minute(s), 2 second(s))
```

Рисунок 2.13 - Результат сборки проекта

Таблица 2.11 - Статистика по выходному файлу проекта

Параметр	Значение
Размер секции .text (Байт)	58639
Размер секции .data (Байт)	136
Размер секции .bss (Байт)	48904
ИТОГО (Байт)	107679
размер файла .elf (Байт)	662640 (647.11 КБ)
размер файла .hex (Байт)	165376
размер файла .bin (Байт)	58788

## 2.5.8 Загрузка исполняемого кода в микроконтроллер

Процедура разгрузки выходного .hex файла в память микроконтроллера с помощью программы-загрузчика «stm32flash»: [35]

- 1) подключить преобразователь «USB-RS485» к разъему «COM.1» ПЛК;
- 2) перевести переключатель «RUN/BOOT» в положение «BOOT»;
- 3) перезапустить ПЛК (кнопка «RESET» или выключить/включить питание);
- 4) запустить программу-загрузчик со следующими атрибутами:

```
stm32flash.exe -w bin/Debug/plc41lrte.elf.hex
               -v -g 0x0 -S 0x08000000 -b 57600 COM5
```

где,

- w bin/Debug/plc411rte.elf.hex — путь к файлу .hex
- g 0x0 – начальный адрес памяти программ
- S 0x08000000 – начальный адрес памяти программ, куда будет записываться исполняемый код (содержимое файла .hex)
- b 57600 – скорость передачи данных (битрейт)
- COM5 – имя последовательного порта ПК в ОС

Процесс загрузки будет отображаться на экране (рисунок 2.14):

```
stm32flash 0.5
http://stm32flash.sourceforge.net/

Using Parser : Intel HEX
Interface serial_w32: 115200 8E1
Version      : 0x31
Option 1     : 0x00
Option 2     : 0x00
Device ID    : 0x0411 (STM32F4xxxx)
- RAM        : 128KiB (8192b reserved by bootloader)
- Flash      : 512KiB (size first sector: 1x16384)
- Option RAM : 16b
- System RAM : 30KiB
Write to memory
Erasing memory
Wrote and verified address 0x08022200 (52.60%)
```

Рисунок 2.14 - Процесс загрузки

5) дождаться завершения загрузки.

## 2.6 Создание описания целевой платформы

При создании нового проекта среде программирования Beremiz в настройках проекта выбирается (из списка) соответствующая «Целевая платформа» - т. е. указывается для какого устройства и его системы исполнения будет разрабатываться управляющая программа.

В Beremiz YAPLC [26] описания целевых платформ находятся в директории Beremiz YAPLC / IDE / yaplctargets (см. п.1.4.1 «Beremiz YAPLC»).

## ЗАКЛЮЧЕНИЕ

В данной выпускной квалификационной работе:

- 1) Выполнен анализ технического задания на разработку встраиваемой системы исполнения для предоставленного отладочного экземпляра аппаратной платформы «ПЛК411»;
- 2) Проведена работа с технической и справочной литературой, по результатам чего был выполнен ОБЗОРНЫЙ РАЗДЕЛ, где кратко рассмотрены: классы программируемых устройств и встраиваемого ПО, функционал ОСРВ, стандарт МЭК-61131-3 и принцип работы среды Beremiz;
- 3) На основе исходных данных задания была разработана функциональная схема контроллера и схема многозадачности его системы исполнения;
- 4) По результатам анализа технического задания был выполнен ОСНОВНОЙ РАЗДЕЛ, где было дано подробное описание работы разрабатываемой системы исполнения (от уровня периферии контроллера до уровня процессов ОСРВ) и сформирована карта адресов регистров данных;
- 5) Выполнено программирование системы исполнения: от создания нового проекта до получения исполняемого файла;
- 6) Создано описание разработанной системы исполнения для среды Beremiz.

Полученный исполняемый файл системы исполнения был загружен в «ПЛК411». Описание системы исполнения было интегрировано в среду Beremiz.

Для проверки работоспособности системы исполнения, в среде Beremiz был создан проект управляющей программы для автоматизации отладочного стенда «Камера термообработки». Стенд включает в себя: управляющий ПЛК411, вентилятор и нагревательный элемент в качестве исполнительных механизмов, датчик температуры как элемент обратной связи. Проект был успешно собран и загружен в контроллер. Результат работы системы: мигание светодиодов ПЛК («RUN» и «NET»), отображение работы системы в режиме реального времени на экране web-ориентированной SCADA (данные передаются по протоколу MODBUS в режиме реального времени) и в окне среды Beremiz (режим отладки).

					27.03.04.2023.203.23.05 ВКР	Лист
Изм.	Лист	№ докум.	Подпись	Дата		115

## БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Лившиц Ю.Е. Программируемые контроллеры для управления технологическими процессами: учебно-методическое пособие и лабораторные работы для студентов всех форм обучения и специальностей «Автоматизация технологических процессов и производств», «Промышленные роботы и робототехнические комплексы», «Программное обеспечение информационных технологий», «Информационные системы и технологии»: в двух частях. / Ю.Е. Лившиц, В.И. Лакин, Ю.И. Монич. - Минск: БНТУ, 2014 г. - Ч.1 — 206 с.
2. Введение в ПЛК: что такое программируемый логический контроллер / Компания "Компэл", г.Москва. - <https://www.compel.ru/lib/95591>
3. Овен ПР110. - <https://docs.owen.ru/product/pr110>
4. Овен ПР200. - <https://docs.owen.ru/product/pr200>
5. Овен ПЛК110. - <https://docs.owen.ru/product/plk110>
6. ПЛК Segnetics Pixel. - <https://ru-segnetics.com/segnetics-pixel>
7. ПЛК SIEMENS S7-400. - <https://www.siemens-pro.ru/components/s7-400.htm>
8. ПЛК ICP DAS iP-8847. - <https://icp-das.ru/products/ip-8447>
9. ПЛК ПРОСОФТ REGUL R500. - <https://prosoftsystems.ru/regul-r500>
10. Распределенная система управления SIEMENS PCS7. - <https://www.novec.ru/catalog/novosibirsk/catalog/siemens/pcs7.pdf>
11. Курниц А. FreeRTOS – операционная система для микроконтроллеров / А. Курниц // Журнал "Компоненты и технологии". - 2011. - Вып.3. - С.109-114.
12. Курниц А. FreeRTOS – операционная система для микроконтроллеров / А. Курниц // Журнал "Компоненты и технологии". - 2011. - Вып.5. - С.97-102.
13. Курниц А. FreeRTOS – операционная система для микроконтроллеров / А. Курниц // Журнал "Компоненты и технологии". - 2011. - Вып.2. - С.96-99.
14. Курниц А. FreeRTOS – операционная система для микроконтроллеров / А. Курниц // Журнал "Компоненты и технологии". - 2011. - Вып.7. - С.23-32.
15. Курниц А. FreeRTOS – операционная система для микроконтроллеров / А. Курниц // Журнал "Компоненты и технологии". - 2011. - Вып.6. - С.98-104.



16. Курниц А. FreeRTOS – операционная система для микроконтроллеров / А. Курниц // Журнал "Компоненты и технологии". - 2011. - Вып.8. - С.132-140.
17. Курниц А. FreeRTOS – операционная система для микроконтроллеров / А. Курниц // Журнал "Компоненты и технологии". - 2011. - Вып.10. - С.93-100.
18. Курниц А. FreeRTOS – операционная система для микроконтроллеров / А. Курниц // Журнал "Компоненты и технологии". - 2011. - Вып.4. - С.96-102.
19. PLCopen. Введение в языки стандарта МЭК-61131-3, 2017 г. -  
[http://www.plcopen.org/pages/tc1\\_standards/iec\\_61131\\_3/](http://www.plcopen.org/pages/tc1_standards/iec_61131_3/)
20. Мидюков А. Beremiz – свободная среда программирования ПЛК / А. Минюков // Электронный журнал "ALT-review". - 2017. -  
<https://www.altlinux.org/Beremiz>
21. Среда программирование Beremiz. Руководство по эксплуатации, 2020 г. -  
[http://www.sm1820.com.ru/files/beremiz/beremiz\\_manual.pdf](http://www.sm1820.com.ru/files/beremiz/beremiz_manual.pdf)
22. Настройка параметров FreeRTOS. / microsin.net. -  
<http://microsin.net/programming/arm/freertos-customisation.html>
23. СТО ЮУрГУ 04-2008 Стандарт организации. Курсовое и дипломное проектирование. Общие требования к содержанию и оформлению / Т.И. Парубочая, Н.В. Сырейщикова, В.И. Гузеев, Л.В. Винокурова. - Челябинск: ЮурГУ, 2008. - 56 с.
24. СТО ЮУрГУ 21-2008 Стандарт организации. Система управления качеством образовательных процессов. Курсовая и выпускная квалификационная работа. Требования к содержанию и оформлению / Н.В. Сырейщикова, А.Е. Шевелев, Е.В. Шевелева. - Челябинск: ЮурГУ, 2008. - 55 с.
25. Beremiz is Free Software for machine automation. Official website. -  
<https://beremiz.org/>
26. Beremiz YAPLC/Nucleron. GitHub: Information and Releases. -  
<https://github.com/nucleron/YAPLC>
27. Customisation FreeRTOSConfig.h - <https://freertos.org/a00110.html>
28. Eclipse Embedded C/C++ Development Tools (GNU MCU/ARM Eclipse) -  
<https://eclipse-embed-cdt.github.io/>

29. IEC 61131-3:2013 Programmable controllers — Part 3, 2012. -  
<https://webstore.iec.ch/publication/4552>
30. MODBUS application protocol specification. V1.1b3, 2012. - C.6-49.
31. Newlib C library for embedded systems - <https://sourceware.org/newlib>
32. STM32F411xC, STM32F411xE. Datasheet - production data. Rev.7: STMicroelectronics, 2017. - C.1, 12, 15, 18-20, 27-29, 35, 38-53, 130-131.
33. STM32F411xC/E advanced Arm-based 32-bit MCUs. Reference manual. RM0383. Rev.3: STMicroelectronics, 2018. - C.92-97, 99, 145-151, 152-155, 165-188, 201-207, 212-227, 352-373, 505-531, 541, 546-547.
34. STM32F4. Description of HAL and low-layer drivers. Rev.7, 2021. - 2123 c.
35. stm32flash. GitHub: Information and Releases. -  
<https://github.com/ARMinARM/stm32flash>

# ПРИЛОЖЕНИЕ А

## КАРТА АДРЕСОВ РЕГИСТРОВ ДАННЫХ

Таблица А.1 - Карта адресов регистров

Регистр	Тип данных	Доступ				
		Beremiz	ModBus			
		адрес	адрес	таблица	порядок байт	ЕЕ
DI						
DI.0 Нормальный: Значение	BOOL	%IX1.0.1.1	0	DISCRETE INPUTS		
DI.1 Нормальный: Значение	BOOL	%IX1.1.1.1	1	DISCRETE INPUTS		
DI.0 Счетчик импульсов: Значение (имп.)	DWORD	%ID1.0.2.1	0	INPUT REGISTERS	1-0 3-2	
DI.1 Счетчик импульсов: Значение (имп.)	DWORD	%ID1.1.2.1	2	INPUT REGISTERS	1-0 3-2	
DI.0 Счетчик импульсов: Уставка (имп.)	DWORD	%MD1.0.2.2	0	HOLDING REGISTERS	1-0 3-2	+
DI.1 Счетчик импульсов: Уставка (имп.)	DWORD	%MD1.1.2.2	2	HOLDING REGISTERS	1-0 3-2	+
DI.0 Счетчик импульсов: Уставка достигнута, флаг	BOOL	%MX1.0.2.3	2	DISCRETE INPUTS		
DI.1 Счетчик импульсов: Уставка достигнута, флаг	BOOL	%MX1.1.2.3	3	DISCRETE INPUTS		
DI.0 Счетчик импульсов: Разрешение уставки	BOOL	%MX1.0.2.4	0	DISCRETE COILS		+
DI.1 Счетчик импульсов: Разрешение уставки	BOOL	%MX1.1.2.4	1	DISCRETE COILS		+
DI.0 Тахометр: Значение (имп./сек)	WORD	%IW1.0.3.1	4	INPUT REGISTERS	1-0	
DI.1 Тахометр: Значение (имп./сек)	WORD	%IW1.1.3.1	5	INPUT REGISTERS	1-0	
DI.0 Тахометр: Уставка (имп./сек)	WORD	%MW1.0.3.2	4	HOLDING REGISTERS	1-0	+
DI.1 Тахометр: Уставка (имп./сек)	WORD	%MW1.1.3.2	5	HOLDING REGISTERS	1-0	+
DI.0 Тахометр: Уставка достигнута, флаг	BOOL	%MX1.0.3.3	4	DISCRETE INPUTS		
DI.1 Тахометр: Уставка достигнута, флаг	BOOL	%MX1.1.3.3	5	DISCRETE INPUTS		

Продолжение таблицы А.1

Регистр	Тип данных	Доступ				
		Beremiz	ModBus			
		адрес	адрес	таблица	порядок байт	ЕЕ
DI.0 Тахометр: Разрешение уставки	BOOL	%MX1.0.3.4	2	DISCRETE COILS		+
DI.1 Тахометр: Разрешение уставки	BOOL	%MX1.1.3.4	3	DISCRETE COILS		+
DI.0: Режим работы	BYTE	%MB1.0.4	6	HOLDING REGISTERS	1-0	+
DI.1: Режим работы	BYTE	%MB1.1.4	7	HOLDING REGISTERS	1-0	+
DI.0: Сброс счетчика, команда	BOOL	%MX1.0.6	4	DISCRETE COILS		
DI.1: Сброс счетчика, команда	BOOL	%MX1.1.6	5	DISCRETE COILS		
DO						
DO.0 Нормальный: Значение	BOOL	%QX2.0.1.1	6	DISCRETE COILS		
DO.1 Нормальный: Значение	BOOL	%QX2.1.1.1	7	DISCRETE COILS		
DO.0 Быстрый: Значение	BOOL	%QX2.0.2.1	8	DISCRETE COILS		
DO.1 Быстрый: Значение	BOOL	%QX2.1.2.1	9	DISCRETE COILS		
DO.0 ШИМ: Длительность (% от периода)	REAL	%QD2.0.3.1	8	HOLDING REGISTERS	1-0 3-2	
DO.1 ШИМ: Длительность (% от периода)	REAL	%QD2.1.3.1	10	HOLDING REGISTERS	1-0 3-2	
DO.0 ШИМ: Период (мсек)	DWORD	%MD2.0.3.2	12	HOLDING REGISTERS	1-0 3-2	+
DO.1 ШИМ: Период (мсек)	DWORD	%MD2.1.3.2	14	HOLDING REGISTERS	1-0 3-2	+
DO.0 ШИМ: Разрешение работы	BOOL	%MX2.0.3.3	10	DISCRETE COILS		
DO.1 ШИМ: Разрешение работы	BOOL	%MX2.1.3.3	11	DISCRETE COILS		
DO.0: Режим работы	BYTE	%MB2.0.4	16	HOLDING REGISTERS	1-0	+

Продолжение таблицы А.1

Регистр	Тип данных	Доступ				
DO.1: Режим работы	BYTE	%MB2.1.4	17	HOLDING REGISTERS	1-0	+
DO.0 Нормальный: Значение безопасного состояния	BOOL	%QX2.0.1.5	12	DISCRETE COILS		+
DO.1 Нормальный: Значение безопасного состояния	BOOL	%QX2.1.1.5	13	DISCRETE COILS		+
DO.0 Быстрый: Значение безопасного состояния	BOOL	%QX2.0.2.5	14	DISCRETE COILS		+
DO.1 Быстрый: Значение безопасного состояния	BOOL	%QX2.1.2.5	15	DISCRETE COILS		+
DO.0 ШИМ: Длительность (% от периода) безопасного состояния	REAL	%QD2.0.3.5	18	HOLDING REGISTERS	1-0 3-2	+
DO.1 ШИМ: Длительность (% от периода) безопасного состояния	REAL	%QD2.1.3.5	20	HOLDING REGISTERS	1-0 3-2	+
AI						
AI.0 Нормальный: Значение	REAL	%ID3.0.1.1	6	INPUT REGISTERS	1-0 3-2	
AI.1 Нормальный: Значение	REAL	%ID3.1.1.1	8	INPUT REGISTERS	1-0 3-2	
AI.0: Режим работы	BYTE	%MB3.0.4	22	HOLDING REGISTERS	1-0	+
AI.1: Режим работы	BYTE	%MB3.1.4	23	HOLDING REGISTERS	1-0	+
Системная информация						
Код ПЛК	WORD	%MW7.1.0	10	INPUT REGISTERS	1-0	
Код варианта исполнения ПЛК	WORD	%MW7.1.1	11	INPUT REGISTERS	1-0	
Номер версии системы исполнения ПЛК	WORD	%MW7.1.2	12	INPUT REGISTERS	1-0	
Дата выпуска системы исполнения ПЛК	WORD	%MW7.1.3	13	INPUT REGISTERS	1-0	
Год выпуска системы исполнения ПЛК	WORD	%MW7.1.4	14	INPUT REGISTERS	1-0	
Состояние системы исполнения ПЛК	WORD	%MW7.1.5	15	INPUT REGISTERS	1-0	

Изм.	Лист	№ докум.	Подпись	Дата

27.03.04.2023.203.23.05 ВКР

Лист

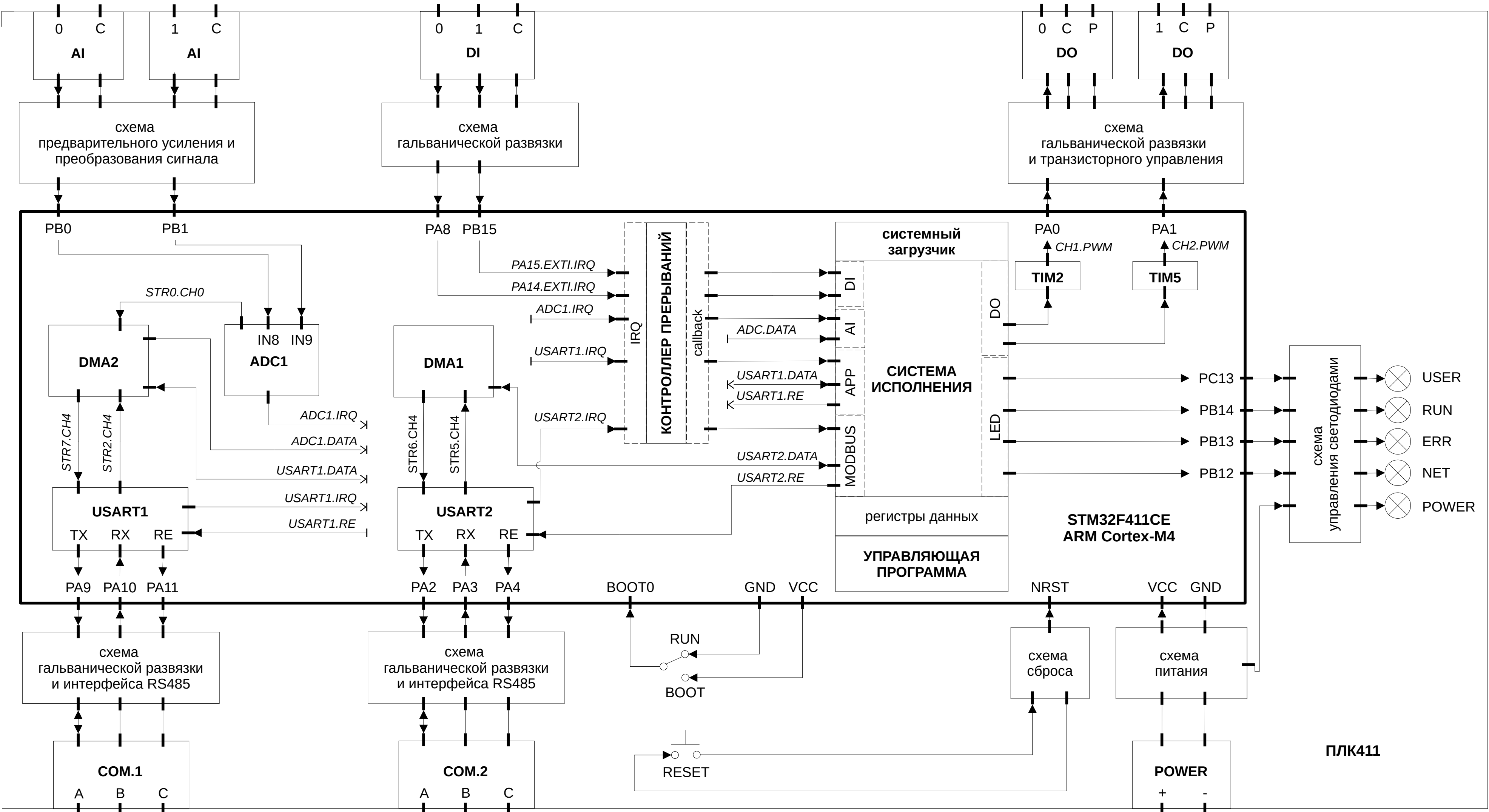
121

## Окончание таблицы А.1

Регистр	Тип данных	Доступ				
		Beremiz	ModBus			
		адрес	адрес	таблица	порядок байт	ЕЕ
Температура ЦПУ ПЛК	REAL	%MD7.2.0	16	INPUT REGISTERS	1-0 3-2	
Системные команды и настройки						
Светодиод USER, команда	BOOL	%MX7.3.0	16	DISCRETE COILS		
Таймер безопасного состояния каналов вывода, команда сброса	BOOL	%MX7.3.2	17	DISCRETE COILS		
Сторожевой таймер ПЛК, команда сброса	BOOL	%MX7.3.3	18	DISCRETE COILS		
Таймер безопасного состояния каналов вывода, время (сек)	WORD	%MW7.4.0	24	HOLDING REGISTERS	1-0	+
Сторожевой таймер ПЛК, время (сек)	WORD	%MW7.4.1	25	HOLDING REGISTERS	1-0	+
Пользовательские регистры (дискретные)						
Регистр (1)	BOOL	%MX8.1.0	19	DISCRETE COILS		
...	BOOL	...	...	DISCRETE COILS		
Регистр (32)	BOOL	%MX8.1.31	50	DISCRETE COILS		
Пользовательские регистры (числовые)						
Регистр (1)	WORD	%MW8.2.0	26	HOLDING REGISTERS	1-0	+
...	WORD	...	...	HOLDING REGISTERS	1-0	+
Регистр (17)	WORD	%MW8.2.16	42	HOLDING REGISTERS	1-0	
...	WORD	...	...	HOLDING REGISTERS	1-0	
Регистр (64)	WORD	%MW8.2.63	89	HOLDING REGISTERS	1-0	

где, ЕЕ – значения (+), хранимые в энергонезависимой памяти ПЛК.

ПРИЛОЖЕНИЕ Б

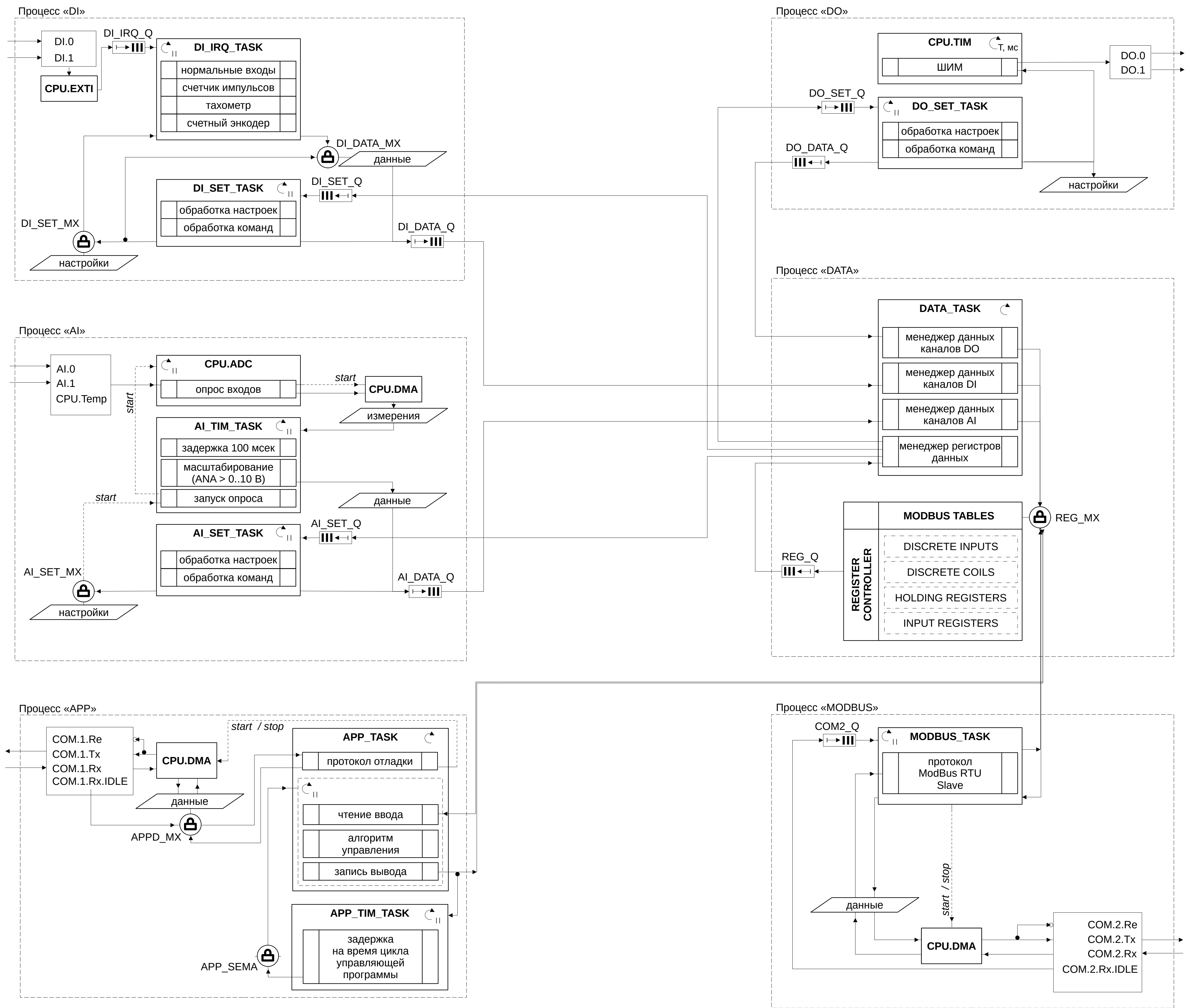


ЛЕГЕНДА

- передача сигнала по стрелке (слева направо) с переходом по метке
- передача сигнала по стрелке (слева направо или справа налево) с переходом по метке
- передача сигнала по стрелке (справа налево) с переходом по метке

					27.03.04.2023.203.23.05 Э2							
					ПЛК411. Схема электрическая функциональная			Лит.		Масса	Масштаб	
Изм.	Лист	№ докум.	Подп.	Дата								
Разраб.												
Пров.												
Конс.								Лист 1		Листов 1		
Н.контр.								ЮУрГУ (НИУ) Кафедра Автоматики				
Утв.												

ПРИЛОЖЕНИЕ В  
СХЕМА МНОГОПОТОЧНОСТИ СИСТЕМЫ ИСПОЛНЕНИЯ ПЛК411



ЛЕГЕНДА

⌚ Т, мс - периодический режим работы (задержка на время Т в миллисекундах)

⌚ - работа в режиме ожидания (ожидание получение данных / прерывание)

🔒 - двоичный семафор или мьютекс

📁 - очередь (стек типа FIFO) с функцией блокировки задачи



# Отчет о проверке на заимствования №1



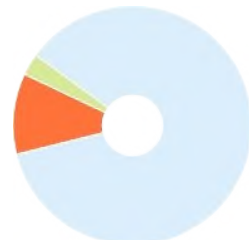
Автор: УНИВЕРИС  
Проверяющий: УНИВЕРИС  
Организация: Южно-Уральский государственный университет  
Отчет предоставлен сервисом «Антиплагиат» - <http://susu.antiplagiat.ru>

## ИНФОРМАЦИЯ О ДОКУМЕНТЕ


№ документа: 330863  
Начало загрузки: 16.05.2023 14:25:27  
Длительность загрузки: 00:00:46  
Имя исходного файла:  
VKP\_2023\_МиЭт-523\_Звездин\_Владимир\_Валерьевич.pdf  
Название документа:  
VKP\_2023\_МиЭт-523\_Звездин\_Владимир\_Валерьевич.pdf  
Размер текста: 176 кБ  
Символов в тексте: 179423  
Слов в тексте: 20758  
Число предложений: 1303

## ИНФОРМАЦИЯ ОБ ОТЧЕТЕ


Начало проверки: 16.05.2023 11:26:14  
Длительность проверки: 00:00:55  
Комментарии: не указано  
Поиск с учетом редактирования: да  
Проверенные разделы: титульный лист с. 1,4, основная часть с. 2-3,5-6,11-119, содержание с. 7-10, библиография с. 120-122, приложение с. 123-128  
Модули поиска: ИПС Адилет, Библиография, Сводная коллекция ЭБС, Интернет Плюс\*, Сводная коллекция РГБ, Цитирование, Переводные заимствования (RuEn), Переводные заимствования по eLIBRARY.RU (EnRu), Переводные заимствования по коллекции Гарант: аналитика, Переводные заимствования по коллекции Интернет в английском сегменте, Переводные заимствования по Интернету (EnRu), Переводные заимствования по коллекции Интернет в русском сегменте, Переводные заимствования издательства Wiley, eLIBRARY.RU, СПС ГАРАНТ: аналитика, СПС ГАРАНТ: нормативно-правовая документация, Медицина, Диссертации НББ, Коллекция НБУ, Перефразирования по eLIBRARY.RU, Перефразирования по СПС ГАРАНТ: аналитика, Перефразирования по Интернету, Перефразирования по Интернету (EN), Перефразированные заимствования по коллекции Интернет в английском сегменте, Перефразированные заимствования по коллекции Интернет в русском сегменте, Перефразирования по коллекции издательства Wiley, Патенты СССР, РФ, СНГ, СМИ России и СНГ, Шаблонные фразы, Модуль поиска "susu", Кольцо вузов, Издательство Wiley, Переводные заимствования




### СОВПАДЕНИЯ

10,88% 


### САМОЦИТИРОВАНИЯ

0% 

### ЦИТИРОВАНИЯ

2,57% 

### ОРИГИНАЛЬНОСТЬ

86,55% 

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

**ОТЗЫВ РУКОВОДИТЕЛЯ  
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ**

Квалификационная работа выполнена

Студентом Звездиным Владимиром Валерьевичем

Факультет Электротехнический / филиал в г.Миассе

Кафедра автоматики Группа МиЭт-523

Направление подготовки 27.03.04 «Управление в технических системах»  
(код) (наименование)

Руководитель Войнов Игорь Вячеславович, д.т.н., профессор кафедры автоматике  
(Ф.И.О., место работы, должность, ученое звание, ученая степень)  
филиал ЮУрГУ в г.Миассе

Работа проверена на заимствование. Оценка оригинальности работы \_\_\_\_\_ %

Характеристика работы студента в период подготовки ВКР:

Студент проявил себя как грамотный специалист: легко обучаем, открыт для новых технологий, способен оценить сложившуюся ситуацию и самостоятельно принять все необходимые решения для разрешения какой-либо проблемы, возникающей в профессиональной деятельности, обладает самоорганизацией и хорошей ритмичностью в работе (выполняет задачи в поставленный срок), умеет работать в команде специалистов различного уровня.

Отмеченные достоинства:

Студент умеет грамотно пользоваться отечественной и иностранной научно-технической и периодической литературой, в том числе патентной. Является уверенным пользователем персонального компьютера. Знает современные аппаратно-программные средства автоматизации, КИП, процессорные архитектуры, протоколы передачи данных, базы данных, системы управления проектами. Знает несколько языков программирования различного уровня. Имеет практический опыт

работы в областях: автоматизация технологических и прикладных процессов,  
встраиваемые системы, приборостроение.

Отмеченные недостатки:

Отсутствуют.

Заключение:

Студент в полной мере освоил образовательную программу согласно п.V образовательного стандарта. Работа в процессе подготовки выпускной квалификационной работы оценена на «Отлично». Возможно присвоение квалификации бакалавра.

Руководитель \_\_\_\_\_ / \_\_\_\_\_ 2023 г.  
(подпись) (ФИО) (дата)

С отзывом руководителя ВКР ознакомлен

Студент \_\_\_\_\_ / \_\_\_\_\_ 2023 г.  
(подпись) (ФИО) (дата)