УТВЕРЖДЕН

RU.СГМА.00541-02 13 01-ЛУ

БЕЗОПАСНЫЙ ЛОКОМОТИВНЫЙ ОБЪЕДИНЕННЫЙ КОМПЛЕКС БЛОК

Программное обеспечение

Ячейка ВС-САУТ-03

Описание программы

RU.СГМА.00541-02 13 01

Листов 33

Инв. № подл.	Подпись и дата	Взам. инв. №	Инв. № дубл.	Подпись и дата

2025

Литера А

АННОТАЦИЯ

Программное обеспечение ячейки ВС-САУТ-03 разработано для работы в составе Безопасного локомотивного объединенного комплекса БЛОК.

Программное обеспечение ВС-САУТ-03 должно отвечать требованиям, представленным в техническом задании на разработку Безопасного локомотивного объединенного комплекса БЛОК.

В настоящем документе представлены условия применения и назначение, логическая структура ПО ПМ субпроцессора и ПМ Artery ячейки ВС-САУТ-03.

ПО ВС-САУТ-03 разработано для микроконтроллеров ATmega128 (программный модуль субпроцессора) и AT32F407 (программный модуль Artery).

СОДЕРЖАНИЕ

Аннотация 2
1. Общие сведения 4
2. Функциональное назначение 5
3. Описание логической структуры 6
3.1. Состав проекта 6
3.2. Общие положения 17
3.3. Конфигурирование ячейки ВС-САУТ-03 в комплекте БЛОК 22
4. Используемые технические средства 23
5. Вызов и загрузка 24
6. Входные и выходные данные 25
6.1. Входные данные 25
6.2. Выходные данные 28
Приложение 1 (справочное) Перечень принятых сокращений 32
Лист регистрации изменений 33

1. ОБЩИЕ СВЕДЕНИЯ

Наименование программы – Программное обеспечение ячейки ВС-САУТ-03 безопасного локомотивного объединенного комплекса БЛОК (далее по тексту – ПО ВС-САУТ-03).

Обозначение программы – RU.СГМА.00541-02.

Язык программирования – Ассемблер (ПМ субпроцессора), C++ (ПМ Artery).

ПО ВС-САУТ-03 предназначено для работы в составе системы безопасного локомотивного объединенного комплекса БЛОК. Программа записана во внутреннюю память микроконтроллеров типа ATmega128A (программный модуль субпроцессора) и AT32F407 (программный модуль Artery).

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

2.1. Программный модуль субпроцессора выполняет функции по приёму и расшифровке информации от приёмника точечного канала, расчету программной траектории движения, передаче данных об объектах в систему автоведения и выдачи команд на управление сбором тяги и торможением.

2.2. Программный модуль Artery реализует функции обеспечения записи, хранения и считывания РПС, обеспечения возможности записи Базы САУТ, передачи данных CAN в модуль AVR, обновления прошивки модуля AVR, управления клапанами КОН и К266.

3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

3.1. Состав проекта

3.1.1. Проект включает в себя 251 файл:

Имя файла	Назначение файла
Программный модуль субпоцессора	
1 ALS.s90	Сбор и обработка информации с автоматической локомотивной сигнализации локомотива
2 Autodrive.s90	Модуль передачи данных в систему автоведения
3 Base_Fl.s90	Подпрограмма чтения и обработки данных, содержащихся во внешней Flash-памяти, по которой производится расчет параметров движения поезда (длины блок-участков, препятствия, профили, ограничения скорости).
4 CRC_32.s90	Программа проверки Flash-памяти в момент запуска исполняемого файла на основе расчета контрольной суммы простым суммированием. При нарушении целостности запуск программы не производится.
5 Command.s90	Сравнение программной траектории с фактической скоростью движения поезда и выбор команды на торможение или разбор тяги
6 DEF_M128.S90	Описание служебных регистров используемого микроконтроллера
7 Def_BE.s90	Описание распределения ОЗУ микроконтроллера для использования в основном тексте программы
8 DeshBase.s90	Подпрограмма чтения и обработки данных, содержащихся во внешней Flash-памяти, по которой производится расчет параметров движения поезда (длины блок-участков, препятствия, профили, ограничения скорости)
9 Diagnost.s90	Контролирует исправность периферийных блоков САУТ-ЦМ/485 (нахождение их в связи, целостность контрольных сумм идентификационной информации и т.д.)
10 EKS.s90	Сбор, передача и обработка информации с внешних систем
11 Falsh_S.s90	Модуль расчета тормозного пути по перепаду скоростей
12 Izm_dP.s90	Сбор информации о давлении в ТМ и ТЦ

Имя файла		Назначение файла
13	MAIN_BE.s90	Основной цикл программы
14	MacroAVR.s90	Макрокоманды для часто используемых программных решений (умножение, деление, переходы и т.д.)
15	MacroBE.s90	Макрокоманды математических операций, используемые в программе для расчета параметров движения, с привлечением дополнительных ресурсов памяти (извлечение корня, умножение, деление и др.)
16	Navi.s90	Модуль работы с данными о географическом положении поезда
17	Param_Lok.s90	Сбор информации о состоянии дискретных сигналов, характеризующих состояние локомотивных цепей
18	RPS_trans.s90	Формирование потока данных формата РПС для регистрации на СН
19	Sv_BE_M128.s90	Организация обмена по линии связи RS-485
20	Torm_kf.s90	Расчет тормозного коэффициента поезда
21	V_pr.s90	Расчёт программных траекторий из условия наличия ограничения скорости, показания АЛС, значения тормозного коэффициента, уклона пути и др.
22	read_cass.s90	Модуль чтения с кассеты и обновления базы данных
23	sv_can_M128.s90	Организация обмена по линии связи CAN
Программный модуль Artery		
1	ArteryAdcDevice.cpp	Драйвер ADC для Artery
2	ArteryAdcDevice.h	Драйвер ADC для Artery
3	ArteryCanDevice.cpp	Драйвер CAN для Artery
4	ArteryCanDevice.h	Драйвер CAN для Artery
5	ArteryCanLoaderIdProvider.cpp	Поставщик идентификационных данных из CAN-загрузчика на Artery
6	ArteryCanLoaderIdProvider.h	Поставщик идентификационных данных из CAN-загрузчика на Artery
7	ArteryFlashMemory.cpp	Внутренняя FLASH-память Artery
8	ArteryFlashMemory.h	Внутренняя FLASH-память Artery

	Имя файла	Назначение файла
9	ArteryGpioInputDevice.h	Драйвер GPIO, сконфигурированного на вход, для Artery
10	ArteryGpioInterrupt.cpp	Драйвер External interrupt/Event controller для Artery
11	ArteryGpioInterrupt.h	Драйвер External interrupt/Event controller для Artery
12	ArteryGpioOutputDevice.h	Драйвер GPIO, сконфигурированного на выход, для Artery
13	ArteryI2cBus.cpp	Драйвер шины I2C для Artery со стороны Мастера
14	ArteryI2cBus.h	Драйвер шины I2C для Artery со стороны Мастера
15	ArteryRebootService.cpp	Сервис перезагрузки процессора для Artery
16	ArteryRebootService.h	Сервис перезагрузки процессора для Artery
17	ArterySleepService.cpp	Сервис реализации задержки на определённое время для Artery
18	ArterySleepService.h	Сервис реализации задержки на определённое время для Artery
19	ArterySpiSyncMaster.cpp	Драйвер шины SPI со стороны мастера для Artery
20	ArterySpiSyncMaster.h	Драйвер шины SPI со стороны мастера для Artery
21	ArterySysTickClock.cpp	Драйвер часов для Artery
22	ArterySysTickClock.h	Драйвер часов для Artery
23	ArterySysTickTimerDevice.cpp	Драйвер таймера SysTick для Artery
24	ArterySysTickTimerDevice.h	Драйвер таймера SysTick для Artery
25	ArteryTimerDevice.cpp	Драйвер таймера для Artery
26	ArteryTimerDevice.h	Драйвер таймера для Artery
27	ArteryUartDevice.cpp	Драйвер UART для Artery
28	ArteryUartDevice.h	Драйвер UART для Artery
29	ArteryWdtDevice.cpp	Драйвер WDT для Artery
30	ArteryWdtDevice.h	Драйвер WDT для Artery
31	DisableInterrupts.h	Реализация DisableInterrupts под Artery
32	GpioInit.h	Инициализация GPIO для Artery
33	CAN2_RX0_InterruptDelegate.h	Делегат, вызываемый по прерыванию CAN2_RX0

	Имя файла	Назначение файла
34	CAN2_SE_InterruptDelegate.h	Делегат, вызываемый по прерыванию CAN2_SE
35	CAN2_TX_InterruptDelegate.h	Делегат, вызываемый по прерыванию CAN2_TX
36	SysTick_InterruptDelegate.h	Делегат, вызываемый по прерыванию SysTick
37	UART4_InterruptDelegate.h	Делегат, вызываемый по прерыванию UART4
38	UART5_interruptDelegate.h	Делегат, вызываемый по прерыванию UART5
39	USART1_InterruptDelegate.h	Делегат, вызываемый по прерыванию USART1
40	Heap.cpp	Простая реализация кучи без удаления для микроконтроллеров
41	Atomic.h	Интерфейс атомарных операций (не могут быть прерваны другим потоком или прерыванием)
42	CanErrorFlags.h	Структура, содержащая флаги ошибок CAN
43	CanMessage.h	Структура, хранящая сообщение CAN
44	Delegate.h	Делегаты, позволяют ссылаться на функцию или метод класса в едином виде
45	DelegateBase.h	Внутренняя часть делегатов
46	InterruptDelegate.h	Тип делегата, который вызывается для обработки прерывания
47	ICanDevice.h	Интерфейс драйвера CAN
48	IClock.h	Часы, показывающие текущее время
49	IEnableDisableDevice.h	Устройство, поддерживающее включение/выключение
50	IGpioInput.h	Интерфейс драйвера GPIO, сконфигурированного на вход
51	IGpioInterrupt.h	Интерфейс драйвера External interrupt/Event controller
52	IGpioOutput.h	Интерфейс драйвера GPIO, сконфигурированного на выход
53	II2cBus.h	Интерфейс драйвера шины I2C со стороны Мастера

	Имя файла	Назначение файла
54	IMemory.h	Интерфейс памяти с произвольным побайтовым доступом
55	INorMemory.h	Интерфейс NOR-памяти
56	IRebootService.h	Интерфейс для перезапуска процессора (программы)
57	ISensorChannelDeviceSide.h	Интерфейс канала измерения аналогового датчика (АЦП и т.п.) со стороны драйвера
58	ISensorDevice.h	Интерфейс драйвера аналогового датчика (АЦП и т.п.)
59	ISleepService.h	Интерфейс для реализации задержки на определённое время
60	ISpiBus.h	Интерфейс шины SPI со стороны мастера
61	ITimerDevice.h	Интерфейс драйвера таймера, реализованного в железе
62	IUartDevice.h	Интерфейс драйвера UART
63	IWatchdogDevice.h	Интерфейс драйвера сторожевого таймера
64	OperationStatus.h	Статус выполнения операции
65	ITimer.h	Интерфейс таймера
66	TimePoint.h	Момент локального времени (метка времени)
67	TimeSpan.h	Промежуток времени, конвертирующийся в разные понятные единицы
68	Uint.h	Возвращает тип, вмещающий N бит
69	NorToAvrBurner.cpp	Прошивает AVR образом ВС-САУТ из NOR-памяти
70	NorToAvrBurner.h	Прошивает AVR образом ВС-САУТ из NOR-памяти
71	SautBaseCopier.cpp	Копирует Базу САУТ с одной памяти на другую
72	SautBaseCopier.h	Копирует Базу САУТ с одной памяти на другую
73	CanToUartConverter.cpp	Ретранслятор CAN-сообщений в UART
74	CanToUartConverter.h	Ретранслятор CAN-сообщений в UART
75	ClassicSpeedCalculator.cpp	Расчёт ограничения скорости в точности по алгоритму САУТ
76	ClassicSpeedCalculator.h	Расчёт ограничения скорости в точности по алгоритму САУТ
77	IBrakes.h	Данные от системы управления тормозами
78	IConfiguration.h	Конфигурация системы и поезда

	Имя файла	Назначение файла
79	IMeasurements.h	Интерфейс для расчёта кривых
80	CharacteristicsProvider.h	Предоставляет характеристики тормозов в зависимости от типа поезда
81	ElectricTrainCharacteristics.h	Характеристики тормозов электрички
82	FreightTrainCharacteristics.h	Характеристики тормозов грузового поезда
83	ICharacteristics.h	Характеристики тормозов поезда
84	PassengerTrainCharacteristics.h	Характеристики тормозов пассажирского поезда
85	Database.cpp	База данных САУТ
86	Database.h	База данных САУТ
87	Object.cpp	Объект в базе данных
88	Object.h	Объект в базе данных
89	NorRpsLogCatalog.cpp	РПС, который пишет из CAN в NOR-буфер и выдаёт этот как каталог логов для LDP
90	NorRpsLogCatalog.h	РПС, который пишет из CAN в NOR-буфер и выдаёт этот как каталог логов для LDP
91	Saut485MonitorModeAvrReset.cpp	Выключает AVR в режиме Монитор по линии САУТ-485
92	Saut485MonitorModeAvrReset.h	Выключает AVR в режиме Монитор по линии САУТ-485
93	SautCanLogMessages.h	Коды лог-сообщений модуля САУТ
94	SautMain.cpp	Главный файл алгоритмов ВС-САУТ
95	SautMain.h	Главный файл алгоритмов ВС-САУТ
96	MemoryTester.cpp	Класс для тестирования NOR-памяти
97	MemoryTester.h	Класс для тестирования NOR-памяти
98	SautTestFirmwareMain.cpp	Главный файл алгоритмов ВС-САУТ
99	SautTestFirmwareMain.h	Главный файл алгоритмов ВС-САУТ
100	ValveController.cpp	Управление клапанами КОН и К266 по команде от ЦО
101	ValveController.h	Управление клапанами КОН и К266 по команде от ЦО
102	main.cpp	Главный файл ВС-САУТ под Artery
103	project.h	Настройки процессора

	Имя файла	Назначение файла
104	CanLogger.cpp	Логер, позволяющий отправить лог в CAN из любого места
105	CanLogger.h	Логер, позволяющий отправить лог в CAN из любого места
106	CanLogMessage.h	Лог-сообщение для передачи через CanLogger
107	SharedCanLogMessages.h	Коды лог-сообщений уровня Shared
108	CanBaudrateCalculator.cpp	Рассчитывает настройки для контроллера CAN
109	CanBaudrateCalculator.h	Рассчитывает настройки для контроллера CAN
110	CanProvider.cpp	Подключает клиентские CanSocket'ы к CAN
111	CanProvider.h	Подключает клиентские CanSocket'ы к CAN
112	CanSocket.cpp	Сокеты CAN, предоставляют возможность нескольким клиентам работать с CAN независимо
113	CanSocket.h	Сокеты CAN, предоставляют возможность нескольким клиентам работать с CAN независимо
114	ModbusMasterRtu.cpp	Modbus-RTU в режиме Мастера
115	ModbusMasterRtu.h	Modbus-RTU в режиме Мастера
116	Saut485Provider.cpp	Работа по линии RS-485 по протоколу САУТ
117	Saut485Provider.h	Работа по линии RS-485 по протоколу САУТ
118	Saut485Socket.h	Сокеты линии САУТ RS-485, предоставляют возможность нескольким клиентам работать с линией
119	Array.h	Массив, с длиной и итератором
120	ArrayEnumerator.h	Итератора массива
121	Event.h	Позволяет нескольким клиентам подписаться на событие
122	IEnumerable.h	Интерфейс перечисляемого объекта
123	IEnumerator.h	Интерфейс перечислителя
124	ListBase.cpp	Типонезависимая часть реализации списка
125	ListBase.h	Типонезависимая часть реализации списка
126	List.h	Реализация односвязного списка без new
127	Property.h	Позволяет подписываться на изменение значения свойства
128	SpiAvrProgrammer.cpp	Прошивальщик AVR по SPI

	Имя файла	Назначение файла
129	SpiAvrProgrammer.h	Прошивальщик AVR по SPI
130	PollingGpioInterrupt.cpp	Реализует IGpioInterrupt, опрашивая по таймеру GpioInput
131	PollingGpioInterrupt.h	Реализует IGpioInterrupt, опрашивая по таймеру GpioInput
132	UartSplitter.cpp	Разделяет один IUartDevice для двух потребителей
133	UartSplitter.h	Разделяет один IUartDevice для двух потребителей
134	AuxResourceIdentification .cpp	Передаёт идентификационную информацию через AUX_RESOURCE по запросу от SYS_DIAGNOSTICS и при старте
135	AuxResourceIdentification .h	Передаёт идентификационную информацию через AUX_RESOURCE по запросу от SYS_DIAGNOSTICS и при старте
136	FudpRebooter.cpp	По протоколу FUDP отвечает идентификацией и перезагружается для передачи управления загрузчику
137	FudpRebooter.h	По протоколу FUDP отвечает идентификацией и перезагружается для передачи управления загрузчику
138	HalfSet.h	Тип полукомплекта
139	Identification.h	Структура идентификационных данных
140	IdProvider.h	Интерфейс поставщика идентификационных данных
141	CanLoaderIdProvider.cpp	Поставщик идентификационных данных из CAN-загрузчика
142	CanLoaderIdProvider.h	Поставщик идентификационных данных из CAN-загрузчика
143	Saut100HIdentification.h	Структура идентификационных данных САУТ, хранящаяся по адресу 100H
144	Saut100HIdProvider.cpp	Поставщик идентификационных данных из области САУТ по адресу 100H
145	Saut100HIdProvider.h	Поставщик идентификационных данных из области САУТ по адресу 100H
146	Unit.h	Модуль в системе БЛОК
147	I2cMemory.cpp	Реализует IMemory для памяти, подключенной по I2C
148	I2cMemory.h	Реализует IMemory для памяти, подключенной по I2C

	Имя файла	Назначение файла
149	SpiMemory.cpp	Реализует IMemory для памяти с постраничной записью, подключенной по SPI
150	SpiMemory.h	Реализует IMemory для памяти с постраничной записью, подключенной по SPI
151	SpiNorMemory.cpp	Драйвер NOR-памяти, подключенной по SPI
152	SpiNorMemory.h	Драйвер NOR-памяти, подключенной по SPI
153	CanPositionChangeSource.cpp	Предоставляет данные об изменении пройденного пути из линии CAN
154	CanPositionChangeSource.h	Предоставляет данные об изменении пройденного пути из линии CAN
155	Distance.h	Расстояние между двумя позициями
156	IPositionChangeSource.h	Интерфейс источника изменения позиции (пройденного пути)
157	Position.h	Позиция (координата X). Разница двух позиций - пройденный путь
158	PositionProvider.cpp	Предоставляет позицию (координату) для запрошенного момента времени в недавнем прошлом
159	PositionProvider.h	Предоставляет позицию (координату) для запрошенного момента времени в недавнем прошлом
160	Saut485PositionChangeSource.cpp	Предоставляет данные об изменении пройденного пути из линии RS-485
161	Saut485PositionChangeSource.h	Предоставляет данные об изменении пройденного пути из линии RS-485
162	ArrayQueue.h	Реализует очередь на памяти, выделенной во вне
163	FifoBuffer.cpp	FIFO-буфер
164	FifoBuffer.h	FIFO-буфер
165	TypelessArrayQueue.cpp	Типонезависимая часть очереди
166	TypelessArrayQueue.h	Типонезависимая часть очереди
167	IQueue.h	Интерфейс очереди
168	NorRingBuffer.cpp	Кольцевой буфер для NOR-памяти
169	NorRingBuffer.h	Кольцевой буфер для NOR-памяти
170	UartStreamWriter.h	Организует очередь для отправки и передаёт из неё по байту в IUartDevice
171	NorVirtualMemory.cpp	Реализует интерфейс IVirtualMemory для работы с NOR-памятью

	Имя файла	Назначение файла
172	NorVirtualMemory.h	Реализует интерфейс IVirtualMemory для работы с NOR-памятью
173	VirtualFileSystem.cpp	Виртуальная файловая система, в которой имя папки определят память, а имя файла - адрес в этой памяти
174	VirtualFileSystem.h	Виртуальная файловая система, в которой имя папки определят память, а имя файла - адрес в этой памяти
175	CrcChecker.cpp	Проверяет контрольную сумму файлов прошивки
176	CrcChecker.h	Проверяет контрольную сумму файлов прошивки
177	FudpClient.cpp	Клиент протокола удалённого обновления прошивки FUDP
178	FudpClient.h	Клиент протокола удалённого обновления прошивки FUDP
179	IFileSystem.h	Интерфейс файловой системы для FUDP
180	IFirmwareSubmodule.h	Субмодуль протоколов удалённого обслуживания FUDP
181	IProgrammerStateProvider.h	Сообщает о том, подключен ли программатор
182	IPropertiesStorage.h	Интерфейс для доступа к свойствам FUDP
183	LoaderStateMachine.cpp	Машина состояний загрузчика
184	LoaderStateMachine.h	Машина состояний загрузчика
185	PredefinedPropertiesStorage.cpp	Выдаёт неизменяемые параметры загрузчика из структуры
186	PredefinedPropertiesStorage.h	Выдаёт неизменяемые параметры загрузчика из структуры
187	CanRemoteSessionInitializer.cpp	Инициатор сессии протоколов удалённого обновления FUDP/LDP по линии CAN
188	CanRemoteSessionInitializer.h	Инициатор сессии протоколов удалённого обновления FUDP/LDP по линии CAN
189	InitIdentification.h	Данные о модуле, необходимые для открытия сессии удалённого обслуживания
190	IRemoteSessionInitializer.h	Инициатор сессии протоколов удалённого обновления FUDP/LDP
191	Submodule.h	Субмодуль протоколов удалённого обслуживания FUDP/LDP
192	CanLdpClient.h	Клиент (модуль) протокола удалённого скачивания логов LDP работающего по CAN через ISO-TP

	Имя файла	Назначение файла
193	ILogCatalog.h	Интерфейс хранилища логов для LDP
194	LdpClient.cpp	Клиент (модуль) протокола удалённого скачивания логов LDP
195	LdpClient.h	Клиент (модуль) протокола удалённого скачивания логов LDP
196	LogSubmodule.h	Субмодуль протоколов удалённого обслуживания LDP
197	Scheduler.h	Планировщик, ставящий задачи в очередь на выполнение
198	Timer.cpp	Высокоуровневый Timer, работающий на Scheduler
199	Timer.h	Высокоуровневый Timer, работающий на Scheduler
200	AuxResourceSimultaneousBoot.cpp	SimultaneousBoot через CAN-сообщения AUX_RESOURCE
201	AuxResourceSimultaneousBoot.h	SimultaneousBoot через CAN-сообщения AUX_RESOURCE
202	SimultaneousBoot.cpp	Обеспечивает одновременную загрузку и перезагрузку полукомплектов
203	SimultaneousBoot.h	Обеспечивает одновременную загрузку и перезагрузку полукомплектов
204	ITextLoggerImplementation.h	Интерфейс для класса, реализующего логирование текстовых сообщений
205	TextLogger.cpp	Логер, позволяющий отправить текстовое сообщение в лог из любого места
206	TextLogger.h	Логер, позволяющий отправить текстовое сообщение в лог из любого места
207	CanDateProvider.cpp	Показывает последнюю дату, переданную по CAN
208	CanDateProvider.h	Показывает последнюю дату, переданную по CAN
209	CanTime.h	Метка времени, которой помечаются CAN-сообщения в системе БЛОК
210	CanTimeSynchronizer.cpp	Часы, подстраивающиеся под общесистемное время системы БЛОК
211	CanTimeSynchronizer.h	Часы, подстраивающиеся под общесистемное время системы БЛОК
212	Clock.cpp	Реализация IClock, увеличивающая время по прерыванию ITimerDevice
213	Clock.h	Реализация IClock, увеличивающая время по прерыванию ITimerDevice

	Имя файла	Назначение файла
214	Date.h	Структура данных, содержащая дату
215	TimeStamped.h	Добавляет метку времени к значению переменной
216	IMessageTransportProtocol.h	Интерфейс протокола транспортного уровня, передающего данные, разделённые на сообщения
217	IsoTp.cpp	Протокол транспортного уровня ISO-TP
218	IsoTp.h	Протокол транспортного уровня ISO-TP
219	IsoTpReceiver.cpp	Приёмник протокола ISO-TP
220	IsoTpReceiver.h	Приёмник протокола ISO-TP
221	IsoTpTransmitter.cpp	Передатчик протокола ISO-TP
222	IsoTpTransmitter.h	Передатчик протокола ISO-TP
223	Bytes.h	Отвечает за перестановку байтов и битов
224	CrcCalculator.h	Считает CRC
225	Mathematics.h	Содержит математические действия
226	Ranged.h	Число с заданным диапазоном
227	ThreadSafe.h	Включает atomic, если для копирования переменной требуется больше одной ассемблерной инструкции
228	UintFor.h	Возвращает тип, вмещающий число N

3.2. Общие положения

3.2.1. При запуске программы субпроцессора выполняется конфигурирование работы микроконтроллера. Выполняется чтение идентификационной информации микросхем flash-памяти и наличие записанной в них базы данных путевых параметров. После этого выполняется установка начальных данных и переход работы в основной цикл программы, в котором выполняется:

- формирование данных для передачи в систему автоведения;
- проверка готовности к работе сменного носителя информации СН;
- проверка целостности базы данных;
- проверка поступления и обработка данных от приёмника точечного канала;

- расчёт пройденного расстояния по информации, поступающий от ПМ БС-ДПС.

При нахождении в основном цикле менее 100 мс ПМ субпроцессора возвращается к его началу.

При нахождении в основном цикле более 100 мс ПМ субпроцессора однократно выполняет «большой» цикл, в котором производится:

- определение текущего маршрута движения и объектов, расположенных на нём;
- последовательное чтение необходимых для работы параметров МПХ с контролем их поступления;
- формирование информации для ПМ Artery для последующей отправки в канал связи CAN;
- определение проследования «виртуального» генератора САУТ;
- определение необходимости сброса информации о развёрнутом маршруте;
- определение необходимости формирования команды на торможение при несанкционированном начале движения;
- определение текущего тормозного коэффициента;
- расчёт текущей программной скорости;
- формирование команды на разбор тяги и служебное торможение в случае превышения значения фактической скорости над значением программной скорости.

После завершения выполнения «большого» цикла ПМ ВС-САУТ-03 переходит к выполнению основного цикла.

В работе ПМ ВС-САУТ-03 предусмотрено выполнение прерываний, в которых выполняется:

- приём и оправка данных в линию связи RS-485;
- приём информации от ПМ Artery, принятой им из канала связи CAN;
- увеличение значения программного счётчика времени.

Обобщенный алгоритм выполнения программы файла «Sv_be.s90» приведен на рисунке 1. Обобщенный алгоритм выполнения программы файла

«Main_be.s90» приведен на рисунке 2. Обобщенный алгоритм выполнения программы файла «V_pr.s90» приведен на рисунке 3.

Файлы «MacroAvr.s90», «DEF_M128.s90», «sv_can_M128.s90», «macrobe.s90» и «def_be.s90» используются во всех ветвях программы и в обобщенном алгоритме не отображены.

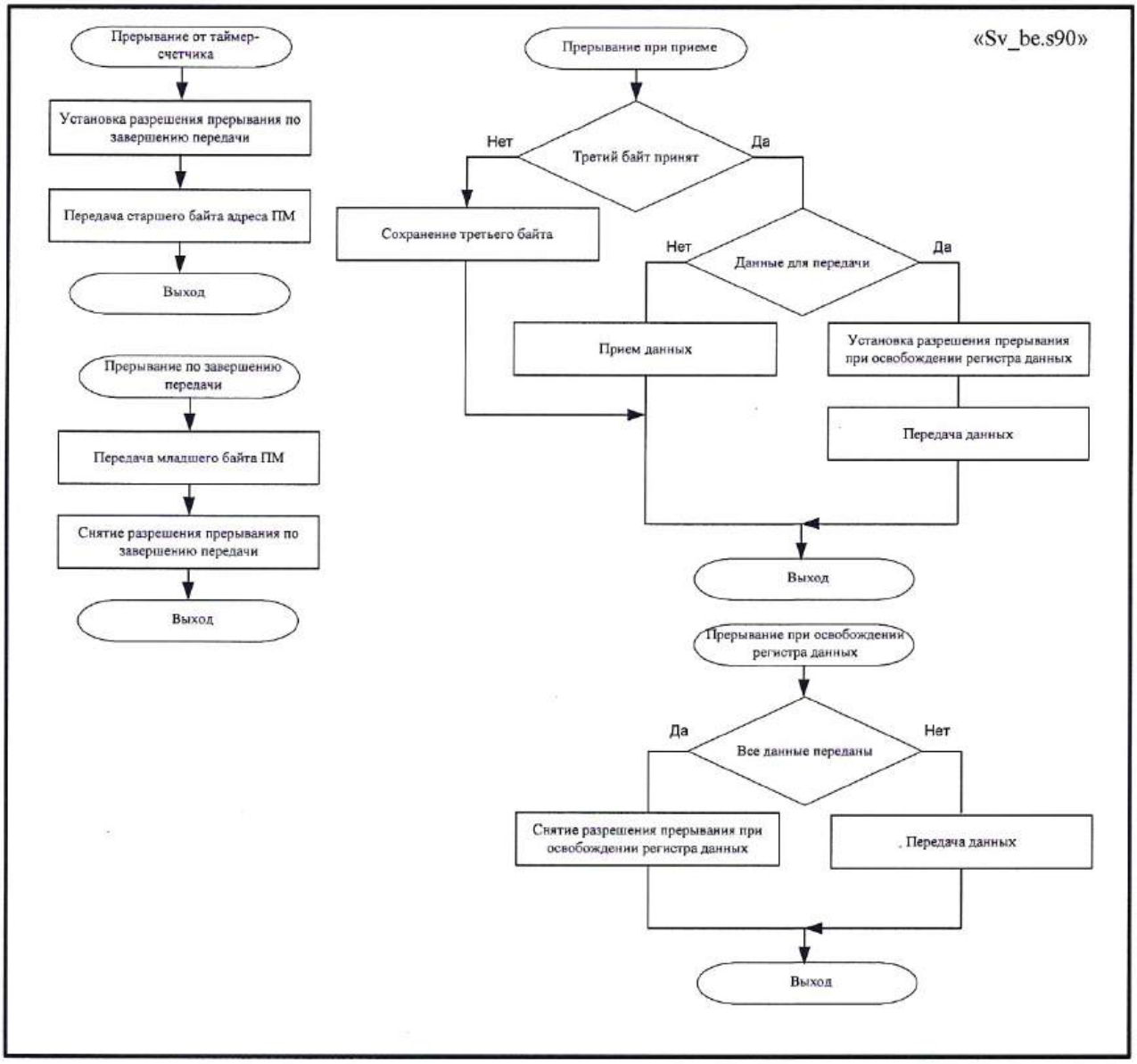

Рисунок 1

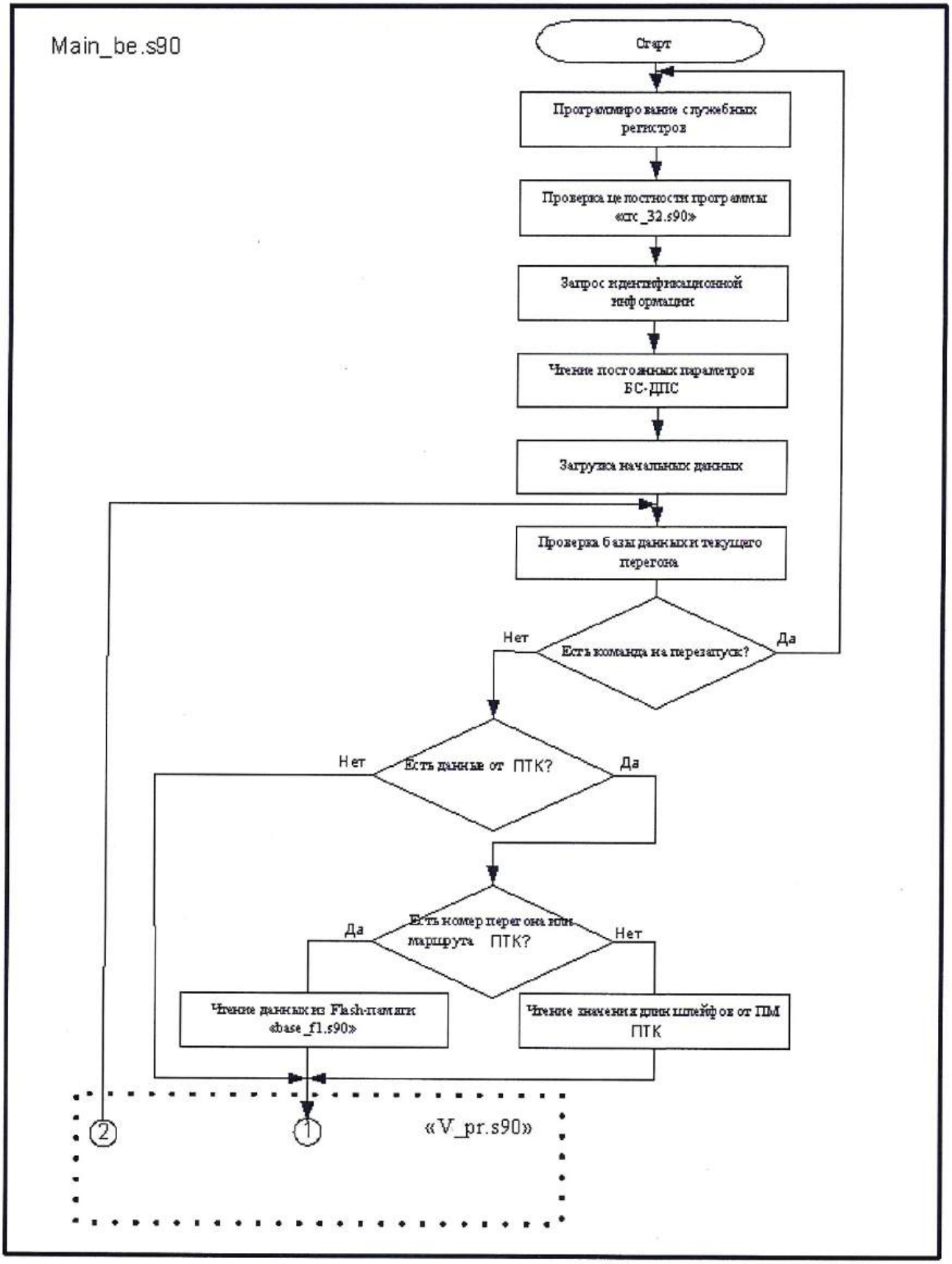
Main_be.s90
Старт
Программирование служебных регистров
Проверка целостности программы «crc_32.s90»
Запрос идентификационной информации
Чтение постоянных параметров БС-ДПС
Загрузка начальных данных
Проверка базы данных и текущего перегона
Есть команда на перезапуск?
Нет
Да
Есть данные от ПТК?
Нет
Да
Есть номер перегона или маршрута ПТК?
Да
Нет
Чтение данных из Flash-памяти «base_f1.s90»
Чтение значений длин шлейфов от ПМ ПТК
2
1
«V_pr.s90»

Рисунок 2

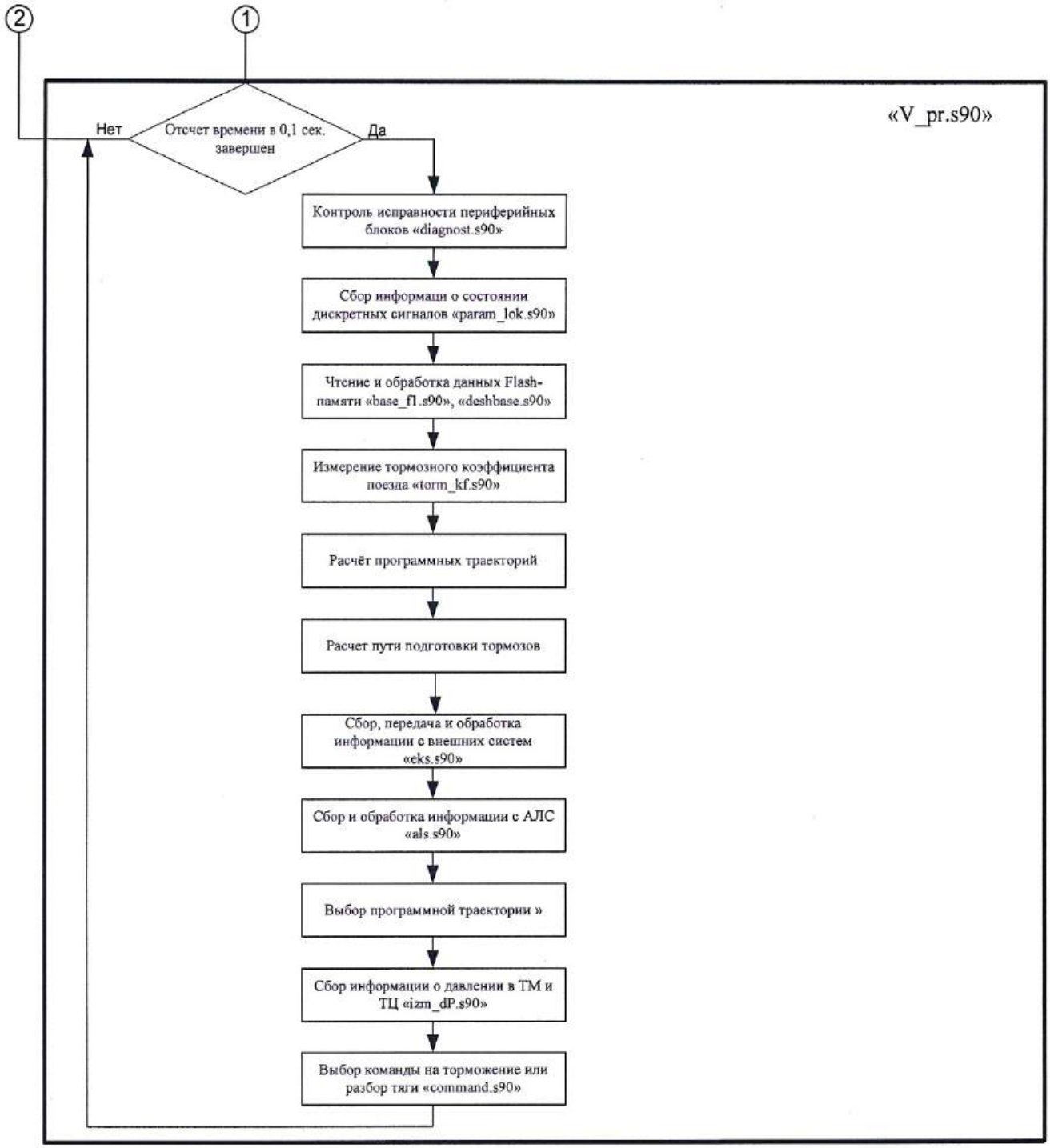

Рисунок 3

Идентификационная информация включает в себя номер блока, дату выпуска, версию ПМ субпроцессора и вариант исполнения блока.

По межпроцессорной линии связи UART программный модуль получает данные РПС от модуля AVR и обеспечивает их сохранение во FLASH-память. Сохранённые данные могут быть считаны по средствам линии CAN в соответствии с протоколом

LDP.

Также, модуль осуществляет ретрансляцию обмена по линии CAN в линию UART, являясь шлюзом для модуля AVR при работе с CAN.

При старте модуль проверяет актуальность прошивки модуля AVR и при необходимости обновляет его.

3.3. Конфигурирование ячейки ВС-САУТ-03 в комплекте БЛОК

3.3.1. Модуль субпроцессора двухканальный, для ввода в конфигурацию ячейки ВС-САУТ-03 производится сравнение данных, получаемых от каждого канала. Конфигурация ячейки ВС-САУТ-03 осуществляется в соответствии с формируемыми субпроцессором сообщениями SAUT_STATE_A и SAUT_STATE_B. Отсутствие или расхождение данных в каналах за время двух последовательных циклов передачи должно приводить к выводу ячейки из активной конфигурации системы. Включение модуля ВС-САУТ в конфигурацию должно происходить после восьми последовательных циклов передачи, в которых не было зафиксировано расхождение или отсутствие данных.

3.4. Программный модуль Artery

3.4.1. По межпроцессорной линии связи UART программный модуль получает данные РПС от модуля AVR и обеспечивает их сохранение во FLASH-память. Сохранённые данные могут быть считаны по средствам линии CAN в соответствии с протоколом LDP.

Также, модуль осуществляет ретрансляцию обмена по линии CAN в линию UART, являясь шлюзом для модуля AVR при работе с CAN.

При старте модуль проверяет актуальность прошивки модуля AVR и при необходимости обновляет его.

4. ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

4.1. Ячейка ВС-САУТ-03 работает только в составе ПО БЛОК.

Используемые микроконтроллеры – ATmega128A (программный модуль субпроцессора) и AT32F407 (программный модуль Artery).

5. ВЫЗОВ И ЗАГРУЗКА

5.1. Программа начинает автоматически выполняться при подаче электропитания на систему БЛОК и отсутствия низкого логического уровня на выводе RESET микроконтроллера, формируемого модулем Artery.

6. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

6.1. Входные данные

6.1.1. Входными для МП субпроцессора являются все данные, принимаемые из линии связи стандарта RS-485, от следующих ПМ (см. файл «Sv_BE_M128.s90»):

1) ПМ блока согласования с автоматической локомотивной сигнализацией:
   - показание локомотивного светофора: зеленый, желтый, красный, красно-желтый, белый («1» - наличие соответствующего сигнала, «0» - отсутствие соответствующего сигнала);
   - тип КПТ («1» - КПТ-7, «0» - КПТ-5);
   - состояние рукоятки бдительности машиниста («1» - РБ нажата, «0»- РБ не нажата).
2) ПМ блока связи с датчиками угла поворота:
   - значение фактической скорости движения подвижного состава;
   - значение фактического ускорения;
   - наличие признака остановки («1» - остановка, «0» - движение);
   - наличие признаков исправности датчиков угла поворота («1» - исправен, «0» - отказ);
   - значения диаметров бандажей;
   - направления вращения датчиков угла поворота («1» - вперед, «0» - назад).
3) ПМ ячейки приёма точечного канала ПТК:
   - признаки наличия частот в шлейфе(19,6 кГц, 27 кГц, 31 кГц);
   - значения амплитуды напряжений сигналов соответствующих частот;
   - значения длин первого и второго шлейфов;
   - значение расстояния от начала первого шлейфа до начала второго;
   - результаты расшифровки кода офм (номер перегона, номер генератора, номер маршрута, тип светофора).
4) ПМ регистратора параметров:
   - значения реального времени и даты.

Также входными для ПМ МП являются следующие сообщения, принимаемые

из линии связи стандарта CAN (см. файл «sv_can_M128.s90»):

1) MCO_STATE:
   - активность УКТОЛ в текущей конфигурации;
   - активность ВС-САУТ в текущей конфигурации.
2) MCO_TORM:
   - код команды управления служебным торможением;
   - актуальная целевая скорость;
   - расстояние до актуальной цели.
3) UKTOL_Got:
   - статус выполнения команды, переданной для УКТОЛ.
4) MSUL_STATE:
   - режим ЭДТ;
   - алгоритм работы САУТ;
   - положение контроллера машиниста.
5) UKTOL_DD:
   - давление в УР;
   - давление в ТЦ;
   - давление в ТМ.
6) UKTOL_UST_DD:
   - установочное давление УКТОЛ.
7) MSUL_DATA1:
   - информация от системы автоведения.
8) SYS_DIAGNOSTICS:
   - общесистемное диагностическое сообщение.
9) SAUT_WAV:
   - команда на воспроизведение звукового сообщения.
10) SYS_KEY:
   - информация о состоянии кнопок «ПОДТЯГ», «ОТПР», «ОС», «К20».
11) BVU_STATE:
   - код принимаемой частоты АЛСН.

12) DISP_STATE:

- положение тумблера Твк.

13) IPD_STATE:

- линейная координата.

14) MM_STATE:

- вид цели электронной карты;
- координата цели.

15) IPD_NEUTRAL:

- признак нейтральной вставки / токораздела;
- расстояние до нейтральной вставки / токораздела.

16) MM_ALT_LONG:

- текущая географическая широта;
- текущая географическая долгота;
- признак достоверности данных о географическом положении.

17) TSKBM_STATE:

- состояние системы ТСКБМ.

18) MCO_LIMITS:

- режим работы – поездной / маневровый / двойная тяга.

19) MM_COORD:

- текущая железнодорожная координата.

20) MPSU_REZ:

- выбранный тип торможения – ПТ / ЭПТ.

21) SYS_DATA:

- общесистемное сообщение о вводимых параметрах.

22) MCO_VELARO:

- признак движения по данным о количестве о свободных блок-участках.

23) MM_DATA:

- признак работы электронной карты в режиме экстраполяции.

24) AMR_STATE:

- признак установленной / снятой кассеты регистрации.

25) ANSW_DB_SAUT_REQ:

- размер занятой области базы данных САУТ на кассете.

26) ANSW_DB_SAUT:

- передача базы данных САУТ с кассеты.

27) MCO_OUT:

- разрешение / запрет передачи CAN-сообщений.

6.1.2. Входными данными для программного модуля Artery являются данные линии CAN, пересылаемые в линию UART, а также данные линии UART, передаваемые в линию CAN.

Также входными данными являются данные РПС, получаемые от модуля AVR.

6.2. Выходные данные

6.2.1. Выходными для ПМ МП являются данные, передаваемые в линию связи стандарта RS-485, для следующих ПМ (см. файл «Sv_BE_M128.s90»):

1) ПМ синтезатора речи пульта машиниста:

- код и порядковый номер фразы.

2) ПМ блока связи с датчиками угла поворота:

- значения длин эталонных блок участков;
- значение фактической скорости движения;
- значение допустимой скорости движения;
- значение расстояния до точки прицельной остановки локомотива;
- значение тормозного коэффициента;
- состояние исполнительных цепей («1» - исполнительные цепи включены, «0» - исполнительные цепи выключены);
- наличие запрета отпуска тормозов («1» - запрет отпуска тормозов, «0» - отпуск тормозов разрешен);
- команды управления торможением и разбором тяги.

Также выходными для ПМ МП являются следующие сообщения, передаваемые в линию связи стандарта CAN (см. файл «sv_can_M128.s90»):

1) SAUT_TORM:

- код команды управления УКТОЛ;
- параметр команды управления УКТОЛ.

2) AUX_RESOURCE_08:

- передача номера версии, подверсии, контрольной суммы.

3) SAUT_STATE :

- состояние исполнительных цепей;
- состояние ЭПК;
- команда на разбор тяги;
- признак движения по базе данных;
- признак движения по перегону;
- запрет отпуска тормозов;
- тормозной коэффициент;
- программная скорость;
- расстояние до цели;
- целевая скорость САУТ.

4) SAUT_DATA1:

- скорость движения фактическая;
- скорость движения допустимая;
- алгоритм работы САУТ – грузовой / пассажирский;
- состояние ЭПК;
- состояние исполнительных цепей;
- текущее показание локомотивного светофора;
- номер перегона;
- тип генератора САУТ;
- номера маршрута и генератора;
- расстояние от выходного генератора САУТ.

5) SAUT_DATA2:

- тип локомотива;
- номер локомотива;

- номер секции;
- тормозной коэффициент;
- код неисправности.

6) SAUT_DATABASE:

- координата события базы данных САУТ;
- код события;
- параметры события.

7) SAUT_DATA3:

- расстояние от выходного генератора САУТ;
- скорость движения фактическая;
- разрядка тормозной магистрали;
- признак движения назад;
- признак запрета отпуска тормозов;
- признак кодирования;
- наличие частоты 31 кГц / 27 кГц / 19,6 кГц;
- признак разгона / замедления;
- величина ускорения / замедления.

8) SAUT_DATA4:

- признак перегон / станция;
- признак боковой / главный путь.

9) SAUT_DATA5:

- расстояние до точки прицельной остановки;
- текущая железнодорожная координата.

10) SAUT_DB_STATE:

- тип объекта базы данных САУТ;
- координата объекта базы данных САУТ.

11) SAUT_READY:

- номер генератора;
- номер маршрута;
- номер пути прибытия;

- номер таблицы АЛС-ЕН;
- железнодорожная координата.

12) SYS_DATA_QUERY:

- запрос параметра.

13) REQ_DB_SAUT:

- начальный адрес чтения информации с кассеты регистрации.

14) ANSW_DB_SAUT:

- команда на чтение информации с кассеты регистрации.

6.2.2. Выходными данными для программного модуля Artery являются записи РПС, отправляемые при считывании.

ПРИЛОЖЕНИЕ 1

(Справочное)

ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ

БЛОК – безопасный локомотивный объединенный комплекс;

ВС-САУТ – ячейка вычисления скорости;

КПТ - кодовый путевой трансмиттер;

КПТ-5 - кодовый путевой трансмиттер, тип 5;

КПТ-7- кодовый путевой трансмиттер, тип 7;

МП – центральный вычислитель;

ОЗУ – оперативное запоминающее устройство;

ОФМ – относительная фазовая манипуляция;

ПО – программное обеспечение;

ПМ – программный модуль;

РБ – рукоятка бдительности машиниста;

ТЦ – тормозные цилиндры;

ТМ – тормозная магистраль;

ЭПК – электропневматический клапан.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	Номер документа	Номер сопроводительного документа и дата	Подпись	Дата
	Измененных	Замененных	Новых	Аннулированных					