УТВЕРЖДЕН

RU.СГМА.00043-01 13 01-ЛУ

БЕЗОПАСНЫЙ ЛОКОМОТИВНЫЙ ОБЪЕДИНЕННЫЙ КОМПЛЕКС

МАСШТАБИРУЕМЫЙ БЛОК-М

Программное обеспечение

Ячейка ПТК-01

Описание программы

RU.СГМА.00043-01 13 01

Листов 24

2024

Литера А

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

АННОТАЦИЯ

В настоящем документе представлены условия применения, назначение и логическая структура программного обеспечения модуля приёмника точечного канала.

СОДЕРЖАНИЕ

1 ОБЩИЕ СВЕДЕНИЯ

Наименование программы – Программное обеспечение ячейки ПТК-01 безопасного локомотивного объединенного комплекса масштабируемого БЛОК-М (далее по тексту ПО ПТК-01).

Обозначение программы - RU.СГМА.00043-01.

Язык программирования – AVR Assembler.

Программное обеспечение ячейки ПТК (далее по тексту ПО ячейки ПТК-01) разработано для работы в составе безопасного локомотивного объединенного комплекса масштабируемого БЛОК-М.

Программное обеспечение ПТК-01 должно отвечать требованиям, представленным в техническом задании на разработку Безопасного локомотивного объединенного комплекса масштабируемого БЛОК-М.

Вычислительный модуль ячейки ПТК-01 выполнен на микроконтроллере AT32F403.

2 ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

ПТК-01 является одним из модулей, входящих в состав ПО БЛОК-М, и выполняет функции по приему и обработке информации от путевых устройств САУТ.

Перечень принятых сокращений приведён в приложении 1.

3 ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

Проект включает в себя 245 файлов:

	Имя файла	Назначение файла
1	GlobalCanId.h	Список дескрипторов CAN
2	ArteryAdcDevice.cpp	Драйвер ADC для Artery
3	ArteryAdcDevice.h	Драйвер ADC для Artery
4	ArteryCanDevice.cpp	Драйвер CAN для Artery
5	ArteryCanDevice.h	Драйвер CAN для Artery
6	ArteryCanLoaderIdProvider.cpp	Поставщик идентификационных данных из CAN-загрузчика на Artery
7	ArteryCanLoaderIdProvider.h	Поставщик идентификационных данных из CAN-загрузчика на Artery
8	ArteryFlashMemory.cpp	Внутренняя FLASH-память Artery
9	ArteryFlashMemory.h	Внутренняя FLASH-память Artery
10	ArteryGpioInputDevice.h	Драйвер GPIO, сконфигурированного на вход, для Artery
11	ArteryGpioInterrupt.cpp	Драйвер External interrupt/Event controller для Artery
12	ArteryGpioInterrupt.h	Драйвер External interrupt/Event controller для Artery
13	ArteryGpioOutputDevice.h	Драйвер GPIO, сконфигурированного на выход, для Artery
14	ArteryI2cBus.cpp	Драйвер шины I2C для Artery со стороны Мастера
15	ArteryI2cBus.h	Драйвер шины I2C для Artery со стороны Мастера
16	ArteryRebootService.cpp	Сервис перезагрузки процессора для Artery
17	ArteryRebootService.h	Сервис перезагрузки процессора для Artery
18	ArterySleepService.cpp	Сервис реализации задержки на определённое время для Artery
19	ArterySleepService.h	Сервис реализации задержки на определённое время для Artery
20	ArterySpiSyncMaster.cpp	Драйвер шины SPI со стороны мастера для Artery
21	ArterySpiSyncMaster.h	Драйвер шины SPI со стороны мастера для Artery

Имя файла		Назначение файла
22	ArterySysTickClock.cpp	Драйвер часов для Artery
23	ArterySysTickClock.h	Драйвер часов для Artery
24	ArterySysTickTimerDevice.cpp	Драйвер таймера SysTick для Artery
25	ArterySysTickTimerDevice.h	Драйвер таймера SysTick для Artery
26	ArteryTimerDevice.cpp	Драйвер таймера для Artery
27	ArteryTimerDevice.h	Драйвер таймера для Artery
28	ArteryUartDevice.cpp	Драйвер UART для Artery
29	ArteryUartDevice.h	Драйвер UART для Artery
30	ArteryWdtDevice.cpp	Драйвер WDT для Artery
31	ArteryWdtDevice.h	Драйвер WDT для Artery
32	DisableInterrupts.h	Реализация DisableInterrupts под Artery
33	GpioInit.h	Инициализация GPIO для Artery
34	ADC1_2_InterruptDelegate.h	Делегат, вызываемый по прерыванию ADC1_2
35	CAN1_SE_InterruptDelegate.h	Делегат, вызываемый по прерыванию CAN1_SE
36	EXINT15_10_InterruptDelegate.h	Делегат, вызываемый по прерыванию EXINT15_10
37	SysTick_InterruptDelegate.h	Делегат, вызываемый по прерыванию SysTick
38	TMR2_GLOBAL_InterruptDelegate.h	Делегат, вызываемый по прерыванию TMR2_GLOBAL
39	UART4_InterruptDelegate.h	Делегат, вызываемый по прерыванию UART4
40	UART5_InterruptDelegate.h	Делегат, вызываемый по прерыванию UART5
41	USART1_InterruptDelegate.h	Делегат, вызываемый по прерыванию USART1
42	USBFS_H_CAN1_TX_InterruptDelegate.h	Делегат, вызываемый по прерыванию USBFS_H_CAN1_TX
43	USBFS_L_CAN1_RX0_InterruptDelegate.h	Делегат, вызываемый по прерыванию USBFS_L_CAN1_RX0
44	Heap.cpp	Простая реализация кучи без удаления для микроконтроллеров

	Имя файла	Назначение файла
45	Atomic.h	Интерфейс атомарных операций (не могут быть прерваны другим потоком или прерыванием)
46	CanErrorFlags.h	Структура, содержащая флаги ошибок CAN
47	CanMessage.h	Структура, хранящая сообщение CAN
48	Delegate.h	Делегаты, позволяют ссылаться на функцию или метод класса в едином виде
49	DelegateBase.h	Внутренняя часть делегатов
50	InterruptDelegate.h	Тип делегата, который вызывается для обработки прерывания
51	ICanDevice.h	Интерфейс драйвера CAN
52	IClock.h	Часы, показывающие текущее время
53	IEnableDisableDevice.h	Устройство, поддерживающее включение/выключение
54	IGpioInput.h	Интерфейс драйвера GPIO, сконфигурированного на вход
55	IGpioInterrupt.h	Интерфейс драйвера External interrupt/Event controller
56	IGpioOutput.h	Интерфейс драйвера GPIO, сконфигурированного на выход
57	II2cBus.h	Интерфейс драйвера шины I2C со стороны Мастера
58	IMemory.h	Интерфейс памяти с произвольным побайтовым доступом
59	INorMemory.h	Интерфейс NOR-памяти
60	IRebootService.h	Интерфейс для перезапуска процессора (программы)
61	ISensorChannelDeviceSide.h	Интерфейс канала измерения аналогового датчика (АЦП и т.п.) со стороны драйвера
62	ISensorDevice.h	Интерфейс драйвера аналогового датчика (АЦП и т.п.)
63	ISleepService.h	Интерфейс для реализации задержки на определённое время
64	ISpiBus.h	Интерфейс шины SPI со стороны мастера
65	ITimerDevice.h	Интерфейс драйвера таймера, реализованного в железе
66	IUartDevice.h	Интерфейс драйвера UART

Имя файла		Назначение файла
67	IWatchdogDevice.h	Интерфейс драйвера сторожевого таймера
68	OperationStatus.h	Статус выполнения операции
69	ITimer.h	Интерфейс таймера
70	TimePoint.h	Момент локального времени (метка времени)
71	TimeSpan.h	Промежуток времени, конвертирующийся в разные понятные единицы
72	Uint.h	Возвращает тип, вмещающий N бит
73	GolayDecoder.cpp	Декодер телеграмм Голея
74	GolayDecoder.h	Декодер телеграмм Голея
75	GolayFrameChecker.h	Реализация проверки фреймов кода Голея
76	IFrameChecker.h	Интерфейс проверки фреймов кода ОФМ
77	IOfmEvent.h	Интерфейс, предоставляющий время между фронтами сигнала ОФМ (должен быть реализован в платформе)
78	OfmReceiver.cpp	Прием кода ОФМ
79	OfmReceiver.h	Прием кода ОФМ
80	RaoReddyFrameChecker.h	Реализация проверки фреймов кода Рао Редди
81	RawTelegramToCanSender.h	Отправляет необработанные данные кодируемого генератора в CAN
82	SautTelegram.h	Телеграмма САУТ
83	TelegramCollector.cpp	Сборщик телеграмм
84	TelegramCollector.h	Сборщик телеграмм
85	TelegramToCanSender.cpp	Отправляет кодируемого шлейфа в CAN
86	TelegramToCanSender.h	Отправляет кодируемого шлейфа в CAN
87	CanGeneratorDetector.cpp	Предоставляет информацию о проезде генератора, взятую из CAN
88	CanGeneratorDetector.h	Предоставляет информацию о проезде генератора, взятую из CAN
89	DetectedToCanSender.cpp	Отправляет события детектирования шлейфа в CAN
90	DetectedToCanSender.h	Отправляет события детектирования шлейфа в CAN
91	GeneratorFrequency.h	Перечень частот напольных генераторов САУТ

Имя файла		Назначение файла
92	IGeneratorDetector.h	Интерфейс детектора напольного генератора САУТ
93	JointGeneratorDetector.cpp	Принимает два IGeneratorDetector'a и выдаёт коллегиальное решение
94	JointGeneratorDetector.h	Принимает два IGeneratorDetector'a и выдаёт коллегиальное решение
95	SensorGeneratorDetector.cpp	Детектор генератора, анализирующий амплитуду огибающей с ISensorChannel
96	SensorGeneratorDetector.h	Детектор генератора, анализирующий амплитуду огибающей с ISensorChannel
97	TestModeAmplitudesSender.cpp	В режиме тестирования передаёт в CAN текущие амплитуды раз в 10 мсек
98	TestModeAmplitudesSender.h	В режиме тестирования передаёт в CAN текущие амплитуды раз в 10 мсек
99	LengthMeasurer.cpp	Рассчитывает длину шлейфов на основании событий начала и конца
100	LengthMeasurer.h	Рассчитывает длину шлейфов на основании событий начала и конца
101	LengthToCanSender.cpp	Отправляет измеренную длину в CAN
102	LengthToCanSender.h	Отправляет измеренную длину в CAN
103	PktObsoleteControl.cpp	Устаревший вариант взаимодействия с сервисной программой Stand.exe
104	PktObsoleteControl.h	Устаревший вариант взаимодействия с сервисной программой Stand.exe
105	PtkCanLogMessages.h	Коды лог-сообщений модуля ПТК
106	PtkFacade.cpp	Базовый класс ПТК
107	PtkFacade.h	Базовый класс ПТК
108	PtkMain.cpp	Главный файл логического модуля ПТК
109	PtkMain.h	Главный файл логического модуля ПТК
110	PtkTestModeSynchronizationUart.cpp	Проверка UART для синхронизации полукомплектов
111	PtkTestModeSynchronizationUart.h	Проверка UART для синхронизации полукомплектов
112	PtkToSaut485Sender.cpp	Выводит результат всей работы ПТК в линию САУТ-485
113	PtkToSaut485Sender.h	Выводит результат всей работы ПТК в линию САУТ-485
114	main.cpp	Главный файл модуля ПТК на Artery

	Имя файла	Назначение файла
115	OfmEvent.cpp	Предоставляет время между фронтами сигнала ОФМ, срабатывая по IGpioInterrupt и измеряя время по IClock
116	OfmEvent.h	Предоставляет время между фронтами сигнала ОФМ, срабатывая по IGpioInterrupt и измеряя время по IClock
117	project.h	Настройки процессора
118	CanLogger.cpp	Логер, позволяющий отправить лог в CAN из любого места
119	CanLogger.h	Логер, позволяющий отправить лог в CAN из любого места
120	CanLogMessage.h	Лог-сообщение для передачи через CanLogger
121	SharedCanLogMessages.h	Коды лог-сообщений уровня Shared
122	CanBaudrateCalculator.cpp	Рассчитывает настройки для контроллера CAN
123	CanBaudrateCalculator.h	Рассчитывает настройки для контроллера CAN
124	CanProvider.cpp	Подключает клиентские CanSocket'ы к CAN
125	CanProvider.h	Подключает клиентские CanSocket'ы к CAN
126	CanSocket.cpp	Сокеты CAN, предоставляют возможность нескольким клиентам работать с CAN независимо
127	CanSocket.h	Сокеты CAN, предоставляют возможность нескольким клиентам работать с CAN независимо
128	ModbusMasterRtu.cpp	Modbus-RTU в режиме Мастера
129	ModbusMasterRtu.h	Modbus-RTU в режиме Мастера
130	Saut485Provider.cpp	Работа по линии RS-485 по протоколу САУТ
131	Saut485Provider.h	Работа по линии RS-485 по протоколу САУТ
132	Saut485Socket.h	Сокеты линии САУТ RS-485, предоставляют возможность нескольким клиентам работать с линией
133	Array.h	Массив, с длиной и итератором
134	ArrayEnumerator.h	Итератор по массиву
135	Event.h	Позволяет нескольким клиентам подписаться на событие
136	IEnumerable.h	Интерфейс перечисляемого объекта

	Имя файла	Назначение файла
137	IEnumerator.h	Интерфейс перечислителя
138	ListBase.cpp	Типонезависимая часть реализации списка
139	ListBase.h	Типонезависимая часть реализации списка
140	List.h	Реализация односвязного списка без new
141	Property.h	Позволяет подписываться на изменение значения свойства
142	SpiAvrProgrammer.cpp	Прошивальщик AVR по SPI
143	SpiAvrProgrammer.h	Прошивальщик AVR по SPI
144	PollingGpioInterrupt.cpp	Реализует IGpioInterrupt, опрашивая по таймеру GpioInput
145	PollingGpioInterrupt.h	Реализует IGpioInterrupt, опрашивая по таймеру GpioInput
146	UartSplitter.cpp	Разделяет один IUartDevice для двух потребителей
147	UartSplitter.h	Разделяет один IUartDevice для двух потребителей
148	Cabin.h	Номер кабины
149	AuxResourceIdentification.cpp	Передаёт идентификационную информацию через AUX_RESOURCE по запросу от SYS_DIAGNOSTICS и при старте
150	AuxResourceIdentification.h	Передаёт идентификационную информацию через AUX_RESOURCE по запросу от SYS_DIAGNOSTICS и при старте
151	FudpRebooter.cpp	По протоколу FUDP отвечает идентификацией и перезагружается для передачи управления загрузчику
152	FudpRebooter.h	По протоколу FUDP отвечает идентификацией и перезагружается для передачи управления загрузчику
153	Saut485IdPage.h	Страница для САУТ-485 с идентификационными данными
154	HalfSet.h	Тип полукомплекта
155	Identification.h	Структура идентификационных данных
156	IdProvider.h	Интерфейс поставщика идентификационных данных
157	CanLoaderIdProvider.cpp	Поставщик идентификационных данных из CAN-загрузчика

	Имя файла	Назначение файла
158	CanLoaderIdProvider.h	Поставщик идентификационных данных из CAN-загрузчика
159	Saut100HIdentification.h	Структура идентификационных данных САУТ, хранящаяся по адресу 100H
160	Saut100HIdProvider.cpp	Поставщик идентификационных данных из области САУТ по адресу 100H
161	Saut100HIdProvider.h	Поставщик идентификационных данных из области САУТ по адресу 100H
162	Unit.h	Модуль в системе БЛОК
163	I2cMemory.cpp	Реализует IMemory для памяти, подключенной по I2C
164	I2cMemory.h	Реализует IMemory для памяти, подключенной по I2C
165	SpiMemory.cpp	Реализует IMemory для памяти с постраничной записью, подключенной по SPI
166	SpiMemory.h	Реализует IMemory для памяти с постраничной записью, подключенной по SPI
167	SpiNorMemory.cpp	Драйвер NOR-памяти, подключенной по SPI
168	SpiNorMemory.h	Драйвер NOR-памяти, подключенной по SPI
169	CanPositionChangeSource.cpp	Предоставляет данные об изменении пройденного пути из линии CAN
170	CanPositionChangeSource.h	Предоставляет данные об изменении пройденного пути из линии CAN
171	CombinedPositionChangeSource.h	Комбинирует информацию от 2 источников изменения позиции
172	Distance.h	Расстояние между двумя позициями
173	IPositionChangeSource.h	Интерфейс источника изменения позиции (пройденного пути)
174	Position.h	Позиция (координата X). Разница двух позиций - пройденный путь.
175	PositionProvider.cpp	Предоставляет позицию (координату) для запрошенного момента времени в недавнем прошлом
176	PositionProvider.h	Предоставляет позицию (координату) для запрошенного момента времени в недавнем прошлом
177	Saut485PositionChangeSource.cpp	Предоставляет данные об изменении пройденного пути из линии САУТ-485
178	Saut485PositionChangeSo	Предоставляет данные об изменении пройден-

	Имя файла	Назначение файла
	urce.h	ного пути из линии САУТ-485
179	ArrayQueue.h	Реализует очередь на памяти, выделенной во вне
180	FifoBuffer.cpp	FIFO-буфер
181	FifoBuffer.h	FIFO-буфер
182	TypelessArrayQueue.cpp	Типонезависимая часть очереди
183	TypelessArrayQueue.h	Типонезависимая часть очереди
184	IQueue.h	Интерфейс очереди
185	NorRingBuffer.cpp	Кольцевой буфер для NOR-памяти
186	NorRingBuffer.h	Кольцевой буфер для NOR-памяти
187	QueueProcessor.h	Автоматически достаёт элементы из очереди и выполняет над ними пользовательскую операцию
188	NorVirtualMemory.cpp	Реализует интерфейс IVirtualMemory для работы с NOR-памятью
189	NorVirtualMemory.h	Реализует интерфейс IVirtualMemory для работы с NOR-памятью
190	VirtualFileSystem.cpp	Виртуальная файловая система, в которой имя папки определят память, а имя файла - адрес в этой памяти
191	VirtualFileSystem.h	Виртуальная файловая система, в которой имя папки определят память, а имя файла - адрес в этой памяти
192	CrcChecker.cpp	Проверяет контрольную сумму файлов прошивки
193	CrcChecker.h	Проверяет контрольную сумму файлов прошивки
194	FudpClient.cpp	Клиент протокола удалённого обновления прошивки FUDP
195	FudpClient.h	Клиент протокола удалённого обновления прошивки FUDP
196	IFileSystem.h	Интерфейс файловой системы для FUDP
197	IFirmwareSubmodule.h	Субмодуль протоколов удалённого обслуживания FUDP
198	IProgrammerStateProvider.h	Сообщает о том, подключен ли программатор
199	IPropertiesStorage.h	Интерфейс для доступа к свойствам FUDP

Имя файла		Назначение файла
200	LoaderStateMachine.cpp	Машина состояний загрузчика
201	LoaderStateMachine.h	Машина состояний загрузчика
202	PredefinedPropertiesStorage.cpp	Выдаёт неизменяемые параметры загрузчика из структуры
203	PredefinedPropertiesStorage.h	Выдаёт неизменяемые параметры загрузчика из структуры
204	CanRemoteSessionInitializer.cpp	Инициатор сессии протоколов удалённого обновления FUDP/LDP по линии CAN
205	CanRemoteSessionInitializer.h	Инициатор сессии протоколов удалённого обновления FUDP/LDP по линии CAN
206	InitIdentification.h	Данные о модуле, необходимые для открытия сессии удалённого обслуживания
207	IRemoteSessionInitializer.h	Инициатор сессии протоколов удалённого обновления FUDP/LDP
208	Submodule.h	Субмодуль протоколов удалённого обслуживания FUDP/LDP
209	ILogCatalog.h	Интерфейс хранилища логов для LDP
210	LdpClient.cpp	Клиент (модуль) протокола удалённого скачивания логов LDP
211	LdpClient.h	Клиент (модуль) протокола удалённого скачивания логов LDP
212	LogSubmodule.h	Субмодуль протоколов удалённого обслуживания LDP
213	Scheduler.h	Планировщик, ставящий задачи в очередь на выполнение
214	Timer.cpp	Высокоуровневый Timer, работающий на Scheduler
215	Timer.h	Высокоуровневый Timer, работающий на Scheduler
216	ISensorChannel.h	Интерфейс канала измерения аналогового датчика (АЦП и т.п.) со стороны клиента
217	SensorChannel.h	Канала измерения аналогового датчика (АЦП и т.п.)
218	AuxResourceSimultaneousBoot.cpp	SimultaneousBoot через CAN-сообщения AUX_RESOURCE
219	AuxResourceSimultaneousBoot.h	SimultaneousBoot через CAN-сообщения AUX_RESOURCE
220	SimultaneousBoot.cpp	Обеспечивает одновременную загрузку и перезагрузку полукомплектов

Имя файла		Назначение файла
221	SimultaneousBoot.h	Обеспечивает одновременную загрузку и перезагрузку полукомплектов
222	ITextLoggerImplementation.h	Интерфейс для класса, реализующего логирование текстовых сообщений
223	TextLogger.cpp	Логер, позволяющий отправить текстовое сообщение в лог из любого места
224	TextLogger.h	Логер, позволяющий отправить текстовое сообщение в лог из любого места
225	CanDateProvider.cpp	Показывает последнюю дату, переданную по CAN
226	CanDateProvider.h	Показывает последнюю дату, переданную по CAN
227	CanTime.h	Метка времени, которой помечаются CAN-сообщения в системе БЛОК
228	CanTimeSynchronizer.cpp	Часы, подстраивающиеся под общесистемное время системы БЛОК
229	CanTimeSynchronizer.h	Часы, подстраивающиеся под общесистемное время системы БЛОК
230	Clock.cpp	Реализация IClock, увеличивающая время по прерыванию ITimerDevice
231	Clock.h	Реализация IClock, увеличивающая время по прерыванию ITimerDevice
232	Date.h	Структура данных, содержащая дату
233	TimeStamped.h	Добавляет метку времени к значению переменной
234	IMessageTransportProtocol.h	Интерфейс протокола транспортного уровня, передающего данные, разделённые на сообщения
235	IsoTp.cpp	Протокол транспортного уровня ISO-TP
236	IsoTp.h	Протокол транспортного уровня ISO-TP
237	IsoTpReceiver.cpp	Приёмник протокола ISO-TP
238	IsoTpReceiver.h	Приёмник протокола ISO-TP
239	IsoTpTransmitter.cpp	Передатчик протокола ISO-TP
240	IsoTpTransmitter.h	Передатчик протокола ISO-TP
241	Bytes.h	Удобные штуки для перестановки байтов и битов
242	CrcCalculator.h	Считает CRC

Имя файла	Назначение файла
243 Mathematics.h	Удобные штуки, связанные с математикой
244 Ranged.h	Число с заданным диапазоном
245 ThreadSafe.h	Включает atomic, если для копирования переменной требуется больше одной ассемблерной инструкции

3.1 Аналоговый канал

3.1.1 Определение начала и конца шлейфа

Программа ПТК-01 непрерывно измеряет 3 напряжения: огибающей сигнала 19 кГц, огибающей сигнала 27 кГц и огибающей сигнала 31 кГц. Измеренные значения усредняются по времени.

При росте величины измеренного напряжения запоминается время, в которое напряжение превысило заданный уровень.

При прохождении порогов формируется событие.

Информация об этом событии передаётся по линии связи с меткой точного времени его возникновения.

3.1.2 Синхронизация полукомплектов

ПТК-01 имеет 2 одинаковых полукомплекта, работающих с одинаковым ПО. Из-за погрешности настройки схем выделения частот значения напряжений огибающих могут различаться между полукомплектами. Это приведёт к различию в определении времени возникновения события и в дальнейшем к различию вычисленных длин шлейфов. Для минимизации этого различия применяется алгоритм синхронизации полукомплектов.

Полукомплект, определив событие, дожидается информации об этом же событии от соседнего полукомплекта и только после этого это событие используется в алгоритме расчёта длины. Началом шлейфа считается последнее из событий заезда на шлейф, а концом шлейфа считается первое из событий

съезда со шлейфа. Таким образом длина шлейфа получается минимальной из возможных, что является наиболее безопасным вариантом.

3.1.3 Вычисление длины шлейфа

Для вычисления длины шлейфа используется информация о пройденном пути из линии связи. Она привязывается ко времени событий для вычисления координат событий. По координатам заезда и съезда вычисляется длина шлейфа.

В момент возникновения события текущая координата не известна. При получении следующего сообщения с пройденным путём из линии связи производится линейная аппроксимация пути за промежуток времени между сообщениями. С её помощью вычисляется координата для того момента времени, в которое произошло событие.

В случае, если получено только событие съезда, а событие заезда отсутствует, длина шлейфа не рассчитывается.

Информация о измеренной длине шлейфа передаётся по линии связи.

3.1.4 Определение вложенных шлейфов

В шлейф генератора с одной несущей частотой может быть вложен шлейф генератора с другой несущей частотой. В этом случае рассчитывается длина обоих шлейфов, а информация о проезде шлейфа передаётся в линию связи только после проезда внешнего шлейфа.

3.2 Цифровой канал

Если во время следования по шлейфу со схемы выделения кодовой последовательности поступает информация, то она обрабатывается алгоритмом ОФМ.

Расшифрованная посылка передаётся по линии связи в след за информацией о проезде шлейфа.

Далее производится декодирование формата телеграмм САУТ. Декодированные данные передаются по линии связи.

3.3 Сервисные функции

3.3.1 Обновление ПО

Помимо основной программы в ПТК-01 находится программа загрузчика, реализующая функцию обновления основной программы по линии связи. При включении питания загрузчик проверяет целостность основной программы и передаёт управление ей. Основная программа, в процессе работы получив по линии связи команду на переход в режим программирования, перезагружается и таким образом передаёт управление загрузчику.

Для обработки команды на переход в режим программирования в формате FUDP, основная программа запрашивает у загрузчика идентификационную информацию о типе и серийном номере ячейки. Эта функция реализуется по-разному в зависимости от типа используемого микроконтроллера в соответствии с документом, описывающем работу с загрузчиком для данного типа микроконтроллера.

3.3.2 Вывод идентификации в линию связи

В ответ на запрос идентификации по линии связи программа передаёт следующую информацию: версию, подверсию и контрольную сумму ПО, а также тип, модификацию и дату производства ячейки.

Эти данные запрашиваются у загрузчика через ту же функцию, что используется при обработке команды на переход в режим программирования.

4 ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

ПТК-01 работает только в составе ПО БЛОК-М.

Используемый микроконтроллер - AT32F403 (ЦП - 240 МГц, ОЗУ – 96 Кб, ПЗУ - 1024 Кб).

5 ВЫЗОВ И ЗАГРУЗКА

Программа начинает автоматически выполняться при подаче электропитания на комплекс БЛОК-М.

6 ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

Входными данными для ПТК-01 являются:

- аналоговые сигналы с фильтров приемников ячейки ПТК-01 канала 19,6 кГц, канала 27 кГц и канала 31 кГц;
- дискретный сигнал со схемы выделения кодовой последовательности;
- данные о пройденном пути, получаемые от модуля БС-ДПС по линиям CAN или RS-485, а также данные о синхронизации времени, получаемые по линии CAN от модуля ЭК или ЦО.

Выходными для ПТК-01 являются данные:

- событие заезда и съезда со шлейфа;
- длина шлейфа;
- данные кодовой посылки.

ПРИЛОЖЕНИЕ 1

(Справочное)

ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ

БЛОК-М – безопасный локомотивный объединенный комплекс масштабируемый;

ОФМ – относительная фазовая манипуляция;

ПО – программное обеспечение;

ПТК – приемник точечного канала;

САУТ - система автоматического управления торможением поездов;

САУТ-ЦМ/485 – локомотивная аппаратура системы автоматического управления торможением поездов.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Подпись	Дата
	изме-ненных	заменен-ных	новых	аннули-рованных				