Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет»
(Национальный исследовательский университет)
Высшая школа электроники и компьютерных наук
Кафедра «Инфокоммуникационные технологии»

Рецензент

_____Дюков А.М.

“____” _____2017г.

Допустить к защите
Руководитель направления

_____ Ю.С. Карманов

“____”_____2017г.

Алгоритмическое и программное обеспечение универсального блока индикации для техники специального назначения

Направление **11.04.02 «Инфокоммуникационные технологии и системы связи»**

*магистерская программа***«Системы мобильной связи»**

ЮУрГУ – М 11.04.02.2017.374.00 ПЗ

Научный руководитель:
_____А.Н. Николаев
“____” ____2017г.

Магистрант:
студент группы КЭ-272
_____А.С.Пустовойтов
“____” ___2017г.

Нормоконтролер:
_____В.Д. Спицына
“____” _____2017г.

Челябинск 2017

РЕФЕРАТ

Пустовойтов А.С.Алгоритмическое и программное обеспечение универсального блока индикации для техники специального назначения. –Челябинск: ЮУрГУ, ВШЭиКН, 2017, 88 с., 19 илл., 14 табл., библиогр. список – 4 наим.

Ключевые слова: программное обеспечение, блок индикации, микроконтроллер,функция, элементы управления дисплеем.

Объектом исследования являются блок индикации БИ04.70 и система безопасности для пожарных машин ОГМ240. Цель работы заключается в реализации программного обеспечения для серийного блока индикации в составе прибора безопасности. Программа написана в среде IAR Embedded Workbench. Для симулирования работы датчиков использовалась программа VirtDat.

Программное обеспечение используется для управления блоком индикации для техники специального назначения.

В ходе выполнения работы были использованы следующие программные продукты: IAR Embedded Workbench, VirtDat, QT.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современных условиях пожарным автолестницам, особенно в городах, зачастую приходиться работать в стесненных условиях, когда невозможно подъехать к горящему объекту на удобное для развертывания лестницы расстояние, что создает проблему выбора места для ее установки. Операторы автолестниц фактически «на глаз» определяют максимально возможное выдвижение лестницы и, как правило, не достигают предельно допустимых вылетов из боязни опрокинуть машину, так как невозможно учесть все факторы, действующие на лестницу. К их числу следует отнести длину выдвижения лестницы, угол подъема, ветровую нагрузку и многое другое. В случае опрокидывания лестницы шансы на спасения у людей, находящихся в горящем здании, падают практически до нуля. Поэтому приходится тратить бесценное время на расчистку места для установки автолестницы ближе к объекту, рискуя при этом жизнями людей, которым требуется помощь.

В отдельных случаях развертывание автолестницы производится операторами неоправданно далеко от объекта, что вызывает сложность управления ее механизмами при развертывании комплекта колен на максимальную длину и необходимость переустановки, что также сопряжено с потерями времени.

В связи с этим является актуальным повышение безопасности и эффективности работы пожарных автолестниц. Эффективным путем снижения их аварийности является использование приборов безопасности, обеспечивающих контроль параметров работы автолестниц и их ограничение на безопасном уровне.

Актуальность разработки программного обеспечения для БИ04.70 обусловлена новыми возможностями системы безопасности ОГМ240-70.30-010-001 (далее по тексту — «ОГМ240» или «прибор безопасности»).

Решение о разработке программного обеспечения для блока индикации БИ04.70 было принято для реализации следующих целей:

— упрощение контроля за рабочими параметрами системы безопасности. Для реализации данной цели будут использоваться виджеты, наглядно демонстрирующие текущие параметры системы;
— возможность демонстрации пространственного положения стрелы;
— переход на более современную элементную базу. Основные элементы, повлиявшие на производительность системы – это мощный 32-разрядный процессор и графический LCD TFT дисплей;
— реализация возможности использования блока в составе систем безопасности для разных объектов техники.

Задачи программиста в разработке программного обеспечения для блока индикации БИ04.70:

— разработка программного обеспечения для пожарной машины АЛ-30: определение способов расчёта рабочих параметров системы, выбор необходимых виджетов для наглядного демонстрирования характеристик системы оператору;
— реализация отображения положения стрелы в рабочей зоне: написание виджета для демонстрации пространственного положения стрелы и отображение рабочей зоны для стрелы.

До недавнего времени такие приборы в России практически не производились.В 2004 году в НПП «Резонанс» был разработан первый прибор безопасности пожарных автолестниц из серии ПБЛ240. В настоящее время различные модификации приборов, в том числе и ОГМ240 устанавливаются на автолестницы производства ОАО «Пожтехника» (г. Торжок Тверской обл.), ООО «Урало-Сибирская пожарно-техническая компания» (г. Миасс Челябинской обл.), ОАО «Казанский электромеханический завод» (ООО «КЭМЗ», г. Казань) и ООО «Техинком» (г. Тверь).

Их основная задача — помочь оператору в управлении механизмами лестницы, не допустить повреждения или опрокидывания машины. Это достигается путем ограничений параметров или режимов работы автолестницы в следующих ситуациях:

— при достижении максимально допустимого вылета лестницы;
— при максимальном и минимальном выдвижении лестницы;
— при достижении максимального или минимального угла наклона лестницы;
— при работе в зоне кабины (для предотвращения повреждения кабины и транспортной стойки лестницы);
— при работе со стороны невыдвинутых опор;
— в случае превышения номинальной грузоподъемности;
— при превышении допустимой скорости ветра;
— в других аварийных или потенциально-опасных ситуациях.

Для определения достижения максимального вылета вычислительному устройству, расположенному в блоке индикации прибора безопасности, необходимо знать геометрические параметры: текущую длину выдвижения и угол наклона лестницы. Для более точного вычисления вылета дополнительно необходимо измерять разность углов наклона корневой секции и вершины лестничного комплекта и на их основании вводить в алгоритм вычисления вылета поправки, связанные с влиянием прогиба лестницы на этот вылет.

Приборы безопасности серии ОГМ240 позволяют реализовать контроль и ограничение нескольких различных значений максимального вылета в зависимости от оснащения лестницы и способа ее установки (наличие люльки или лифтовой системы, степени выдвижения выносных опор и т. д.).

Оснащение автолестницы датчиками угла наклона и длины стрелы позволяет также организовать защиту механизмов выдвигания-задвигания

и подъема-опускания лестницы от повреждений при достижении ими предельных положений.

Благодаря установке датчика азимута реализуется функция защиты кабины и транспортных стоек от повреждений при неосторожной работе лестницей. Дополнительно датчик азимута дает возможность выполнить защиту автолестницы от опрокидывания при попытках проведения работ со стороны невыдвинутых опор.

При этом в приборе безопасности ОГМ240 учитывается, что полный запрет работы в этой зоне является неоправданным, если у автолестницы хватает запаса устойчивости для разворота в рабочую зону из транспортного положения через борт с невыдвинутыми опорами с минимальным вылетом при полностью сложенной лестнице. Это значительно облегчает работу в стесненных условиях.

Интеграция прибора безопасности и системы пропорционального электрогидравлического управления позволяет:

— уменьшать скорость лестницы при приближении к зоне блокировки, что исключает динамические нагрузки на лестницу при резком останове;
— осуществлять блокирование средствами штатных механизмов, т.е. системе управления, имеющей информацию о необходимости ограничения движения, достаточно не выдавать на выход сигнал включения соответствующего рабочего клапана системы.

Данный способ в настоящее время является самым дорогим в реализации из-за использования пропорциональных электромагнитов управления, однако обеспечивает наиболее оптимальные, с точки зрения плавности и безопасности работы, режимы работы автолестницы.

Кроме функций ограничения параметров работы в аварийно-опасных ситуациях, современный прибор безопасности может:

— управлять системой бокового выравнивания как в ручном, так и в автоматическом режимах работы;
— обеспечивать двустороннюю полудуплексную или дуплексную голосовую связь основного пульта управления с вершиной лестницы;
— обеспечивать индикацию продольного и поперечного угла наклона шасси;
— выполнять функции счетчика времени наработки (моточасов) основного привода лестницы;
— производить индикацию текущих параметров автолестницы на дополнительном пульте управления;
— выводить диагностическую информацию оператору, например, давление масла в двигателе и (или) в гидравлической системе механизмов лестницы, температуру охлаждающей жидкости двигателя, информацию о совмещении ступеней комплекта колен и т.д.

Не вызывает сомнений, что с течением времени объем функций, выполняемых приборами и системами безопасности, будет возрастать. Большинство их, в конечном счете, будет интегрировано с пропорциональным электрогидравлическим управлением автолестниц, что позволит выйти на более высокий уровень обеспечения их безаварийности и эффективности применения.

Для реализации поставленных задач данная работа была разбита на 2 части. В первой части изучается опыт конкурирующих компаний и аппаратная часть платформы, на которой будет установлено программное обеспечение. Во второй части работы приводятся алгоритмы и описываются функции, выполнение которых должно быть реализовано с помощью программного обеспечения.

1 Аппаратная часть универсального блока индикации

В данной главе приводится сравнительный анализ блока индикации БИ04.70 и его аналогов, рассматривается аппаратная часть, а именно основные элементы, влияющие на производительность блока индикации, а также датчики, входящие в состав системы безопасности, данные с которых необходимо обрабатывать.

1.1 Описание блока БИ04.70

Блок индикации БИ04.70 применяется в составе систем СБУК на автомобильных кранах, крана с решетчатой стрелой и гидравлическим приводом, а также в составе приборов безопасности ОГМ240 для пожарных машин. Изображение блока индикации представлено на рисунке 1. Его основные характеристики содержатся в таблице 1.

В составе автомобильного крана с гидравлическим приводом система СБУК обеспечивает плавное управление гидравлическими исполнительными механизмами крана и его защиту от повреждения и опрокидывания. В составе автомобильного подъемника с гидравлическим приводом система СБУК обеспечивает плавное управление гидравлическими механизмами автогидроподъемника, его защиту от повреждения и опрокидывания.

СБУК выполняет функции прибора безопасности в соответствии с требованиями ПБ 10-382-00 Ростехнадзора.

Основные особенности:

— графический дисплей повышенной яркости с высоким разрешением;
— мощный 32-разрядный процессор;
— кнопки с подсветкой;
— закаленное стекло, устойчивое к механическим повреждениям;

— аксессуары для встраиваемого монтажа или установки на поворотном кронштейне.

Панель оператора БИ04.70 предназначена для отображения информации в электронных системах строительно-дорожных машин.

Содержит графический TFT-дисплей размером 7" с возможностью отображения 262114 цветов и набор проводных интерфейсов (USB, RBus).

Для взаимодействия с оператором в диалоговом режиме панель содержит 6 кнопок, назначение которых задается программно в зависимости от режима работы машины и панели.

Имеет последовательный интерфейс связи USB 2.0 для считывания данных встроенного регистратора параметров с помощью флэш-накопителей.

Рисунок 1— Пример применения панели в системе безопасности грузоподъемного крана

Таблица 1 — Основные параметры блока индикации БИ04.70

Параметр	Значение
Графический дисплей: тип и формат размер разрешение количество отображаемых цветов	 графический LCD TFT, 16:9 7" (152,4 × 91,4 мм) 800×480 262114

Продолжение таблицы 1

Параметр	Значение
Последовательные интерфейсы связи	USB 2.0, RBus
Органы управления	6 кнопок с подсветкой, переключатель на 2 положения
Звуковая сигнализация	динамический излучатель на задней стенке
Напряжение питания, В	от 8 до 32
Электрическое соединение	20 контакный разъём
Потребляемая мощность, Вт, не более	3,5
Диапазон температур, °С рабочих предельных	 от минус 40 до +55 от минус 50 до +65
Степень защиты от внешних воздействующих факторов по ГОСТ 14254	IP67
Допустимые вибрационные нагрузки, не более максимальное ускорение, м/с2 в диапазоне частот, Гц	 50 от 50 до 200
Допустимые ударные нагрузки, м/с2, не более	10
Габаритные размеры, мм, не более:	178 × 132 × 27
Масса, кг, не более	0,45

1.2 Описание аналогов БИ04.70

Описание блока БИ04.43

Внешний вид блока индикации БИ04.70 представлен на рисунке 2 . Основные параметры блока индикации БИ04.43 содержит таблица 2.

Основные особенности

- графический ЖК-дисплей с разрешением 202×32;
- питание от сети переменного тока напряжением 220 В;
- одновременное отображение информационных сообщений и рабочих параметров;
- считывание данных регистратора и загрузка в блок индикации параметров крана с помощью SD-карты;
- силовой выход с током коммутации 3 А;
- обработка двух дискретных сигналов постоянного (до 540 В) или переменного (380 Предназначен для работы в качестве центрального управляющего и вычислительного блока прибора безопасности для кранов мостового типа.

Используется для ввода режимов работы прибора и вывода информации о работе крана, а также управления сигналами блокировки.

Рисунок 2— Пример применения панели в системе безопасности грузоподъемного крана

Предназначен для работы в качестве центрального управляющего и вычислительного блока прибора безопасности для кранов мостового типа.

Используется для ввода режимов работы прибора и вывода информации о работе крана, а также управления сигналами блокировки.

Таблица 2— Основные параметры блока индикации БИ04.43

Параметр	Значение
Элементы индикации: Дисплей световые табло светодиодные индикаторы	 ЖК, графический (202×32) 2 (желтое, красное) 27
Интерфейс связи	RBus
Органы управления	8 кнопок на лицевой панели
Звуковая сигнализация	динамический излучатель на задней стенке
Напряжение питания, В	от 8 до 32
Электрические соединения питание линия связи с датчиками входы и выходы считывание регистратора параметров	 вилка FQ14-4ZP вилка AMP Superseal 1,5 (4 контакта) вилка FQ14-4ZP разъем Secure Digital
Напряжение питания, В	~220±10%
Потребляемый ток, А, не более	0,1
Количество обрабатываемых дискретных сигналов переменного (380 В) и постоянного (от 150 до 540 В) тока	2

Продолжениетаблицы2

Параметр	Значение
Параметры силового выхода: ток коммутации, А, не более напряжение коммутации, В, не более	 3 380
Диапазон температур, °С рабочих предельных	 от минус 40 до +55 от минус 50 до +65
Степень защиты от внешних воздействующих факторов по ГОСТ 14254-96	IP54
Допустимые вибрационные нагрузки, не более максимальное ускорение, м/с2 в диапазоне частот, Гц	 50 от 50 до 200
Допустимые ударные нагрузки, м/с2, не более	100
Габаритные размеры, мм, не более:	175 × 160 × 96
Масса, кг, не более	1,3

Блок индикации vSCALE C3 HIRSCHMANN

Внешний вид блока индикации vSCALE C3 HIRSCHMANN представлен нарисунке 3. Основные параметры блока содержиттаблица 3.

Рисунок 3— Блок индикации vSCALE C3 HIRSCHMANN

Таблица 3— Основные параметры блока индикации vSCALE C3 HIRSCHMANN

Параметр	Значение
Элементы индикации: Дисплей Тип дисплея светодиодные индикаторы	 7” (800x480 pixels) сенсорный
Интерфейс связи	Ethernet, CAN, USB
Органы управления	Сенсорный дисплей
Звуковая сигнализация	динамический излучатель
Напряжение питания, В	9…36 В
Диапазон температур, °С	от минус 30 до +70

Продолжение таблицы 3

Параметр	Значение
Степень защиты от внешних воздействующих факторов по ГОСТ 14254-96	IP66
Габаритные размеры, мм, не более:	223 × 139 × 64

Сенсорная логическая панель LP-S070 от компании Autonics

Сенсорная логическая панель LP-S070 является модифицированным поколением графических панелей компании Autonics. Внешний вид логической панели представлен нарисунке 4. Основные параметры блока содержиттаблица 4. Большее число функции и полноцветный сенсорный дисплей позволяют более широко применять логическую панель LP-S070.

Встроенный в устройство программируемый логический контроллер позволяет выполнять широкий круг задач с помощью панелей серии LP-S070. Аналоговый сенсорный жидкокристаллический дисплей диагональю 7 дюймов способен передавать 16,7 млн цветов в высоком контрасте и обладает большими возможностями визуализации информации.

Помимо стандартных интерфейсов подключения сенсорная панель LP-S070 поддерживает подключение через Ethernet и USB, что позволяет широко применять устройство. При этом панель при необходимости собирает и регистрирует различные данные благодаря наличию соответствующей функции регистрации данных.

Основные технические параметры сенсорной логической панели LP-S070

Логические панели серии LP-S070 с сенсорным управлением имеют четыре модели. Основное отличие, важное при выборе конкретной модели,

заключается в наличии или отсутствии различных разъемов и интерфейсов подключения. Универсальность панелей позволяет при необходимости просто заменить одно устройство на другое в случае изменения требований к подключению.

Таблица 4 – Основные параметры блоков индикации компании Autonics

Модели устройств	LP-S070-T9D6-C5T	LP- S070-T9D6-C5R	LP- S070-T9D7-C5T	LP- S070-T9D7-C5R
Диагональ дисплея	7 дюймов			
Тип сенсорного дисплея	Цветной дисплей технологии LCD TFT			
Количество цветов	16 777 216 цветов			
Используемый источник питания	24В=			
Интерфейс подключения	По одному порту типа RS232C и RS422, USB-хост, USB-устройство, Ethernet		Два порта типа RS232C, USB-хост, USB-устройство, Ethernet	
Модуль	«Все в одном»			
Число входов и выходов устройства	16 модулей входа и 16 модулей выхода			
Разъемы ввода/вывода	Разъем модуля для ввода-вывода	Разъем плоского кабеля	Разъем модуля для ввода-вывода	Разъем плоского кабеля

Рисунок 4 – Сенсорная логическая панель LP-S070 от компании Autonics

Назначение

Сенсорная панель LP-S070 может применяться для работы с устройствами промышленной автоматики в рамках автоматических систем управления в различных отраслях современной промышленности. Особенно устройство подойдет для тех случаев, когда необходимо организовать человеко-машинный интерфейс для управления оборудованием и отслеживания информации о работе системы и оборудования в реальном времени. Также серия LP-S070 может быть интегрирована в работу при необходимости управления портативными контрольно-измерительными приборами.

1.3 Анализ БИ04.70 и его аналогов

В сравнение со своим предшественником БИ04.43 блок индикации БИ04.70 обладает рядом преимуществом. Графический LCD TFT дисплей позволяет наглядно отображать информации о состоянии оборудования:

- реализованы виджеты для отображения вылетов, углов крена, загрузки;
- страницы со служебной информацией позволяют просматривать основную информацию о системах;

— можно оценить значения и работоспособность всех датчиков, входящих в состав системы безопасности;
— реализована возможность отображения положения стрелы в зоне работы.

Недостатки зарубежного аналога компании «Hirschmann» по отношению к БИ04.70:

— высокая стоимость системы безопасности;
— невозможность работать при температурах ниже 30 градусов по Цельсию;
— необходимость русификации пользовательского интерфейса.

Преимущества блока индикации компании «Hirschmann»:

— сенсорный дисплей упрощает взаимодействие с прибором безопасности;
— реализовано взаимодействие по Ethernet.

vSCALE C3 HIRSCHMANN является высокотехнологичным блоком индикации, но он не приспособлен к российским природным условиям и не выдерживает ценовой конкуренции.

Блок индикации компании Autonics обладает теми же преимуществами, что и блок индикации компании Hirschmann. Данный блок не подходит для установки на строительную и пожарную технику, так как разрабатывался для устройств промышленной автоматики в рамках автоматических систем управления в различных отраслях современной промышленности.

1.4 Схемотехнические решения для блока индикации БИ04.70

Данный подраздел описывает основные схемотехнические решения, повлиявшие на производительность системы безопасности.

Краткое описание применяемого процессора stm32F2x

В таблице 5 отображены основные параметры контроллера.

— производительность 150DMIPS при 120MHz:

1 Adaptive Real Time “ART” Акселератор позволяет исполнять код из flash с 0 WS на 120МГц, обеспечивая STM32 F - 2 одним из лидеров по производительности среди Cortex M3
2 Значительно улучшенная 32 – битная матрица шин AHB для более эффективного одновременного использования периферийных устройств при обмене данными.

— память:

1 До 1Мб высоконадежной Flash - памяти
2 128Кб RAM
3 Ядро Cortex-M3 с MPU и ETM
4 Flexible Static Memory Interface длявнешнего LCD, SRAM, PSRAM, NOR и NAND flash, CompactFlash до 60МГц

— новая периферия:

1 USB OTG High speed 480Мбит/с с интерфейсом ULPI
2 Camera interface
3 Шифрование: DES, 3DES, AES 256-бит, SHA-1 hash-процессор

Таблица 5– Основные параметры процессора stm32F2x

Параметр	Значение
ЦПУ: Ядро	Cortex-M3
ЦПУ: F,МГц	от 0 до 120
Память: Flash,КБайт	128

Продолжение таблицы 5

Параметр	Значение
Память: RAM,КБайт	64
I/O (макс.),шт.	51
Таймеры: 16-бит,шт	14
Таймеры: RTC	Нет
Интерфейсы: UART,шт	2
Интерфейсы: SPI,шт	3
Интерфейсы: I2C,шт	3
Интерфейсы: USB,шт	1
Интерфейсы: CAN,шт	2
Аналоговые входы: Разрядов АЦП,бит	12
Аналоговые входы: Каналов АЦП,шт	16
Аналоговые выходы: Разрядов ЦАП,бит	12

Описание контроллера SSD1963

Контроллер SSD1963 фирмы SOLOMON SYSTECH является универсальным контроллером TFT дисплеев, с разрешением до 864x480 и цветовым разрешением до 24 бит/пиксель. Контроллер снабжён встроенной статической памятью объёмом в 1215 кБ, а также ФАПЧ генераторами развёртки и т. п., что позволяет использовать минимумом внешней обвязки.

Внешний интерфейс контроллера может быть двух видов - 6800 и 8080. Оба этих интерфейса широко расписаны в технической литературе. Для удобства и по причине того, что интерфейс 8080 поддерживается аппаратно STM32F205, поэтому был выбран именно он.

Ширина данных внешнего интерфейса может быть 8/16/24 бит. Более подробная информация описана в документации на контроллер. От ширины интерфейса не зависит настройка дисплея, поэтому инициализация происходит абсолютно одинаково. Однако в процессе инициализации необходимо настроить ширину данных изображения которые будет принимать контроллер от МК/МП в процессе работы.

Описание датчиков, входящих в состав прибора безопасности для АЛ-30

Датчики перемещения серии ДДС

В таблице 6отображены основные параметры датчика перемещения.

Основные особенности:

— измерение приращения длины до 50 м;
— автоматическое сматывание провода посредством внутренней пружины;
— встроенный датчик угла наклона;
— цифровой интерфейс передачи данных;
— наличие дискретного входа.

Используется в системах безопасности строительно-дорожной техники и предназначен для измерения длины и угла наклона телескопической стрелы грузоподъемного крана.Основные технические данные содержатся в таблице таблице 6. Сборочный чертеж датчика представлен на рисунке 5.

Выполняемые функции

— измерение линейного перемещения прямолинейно движущихся объектов;

— подвод электрической энергии к прямолинейно перемещающимся объектам;

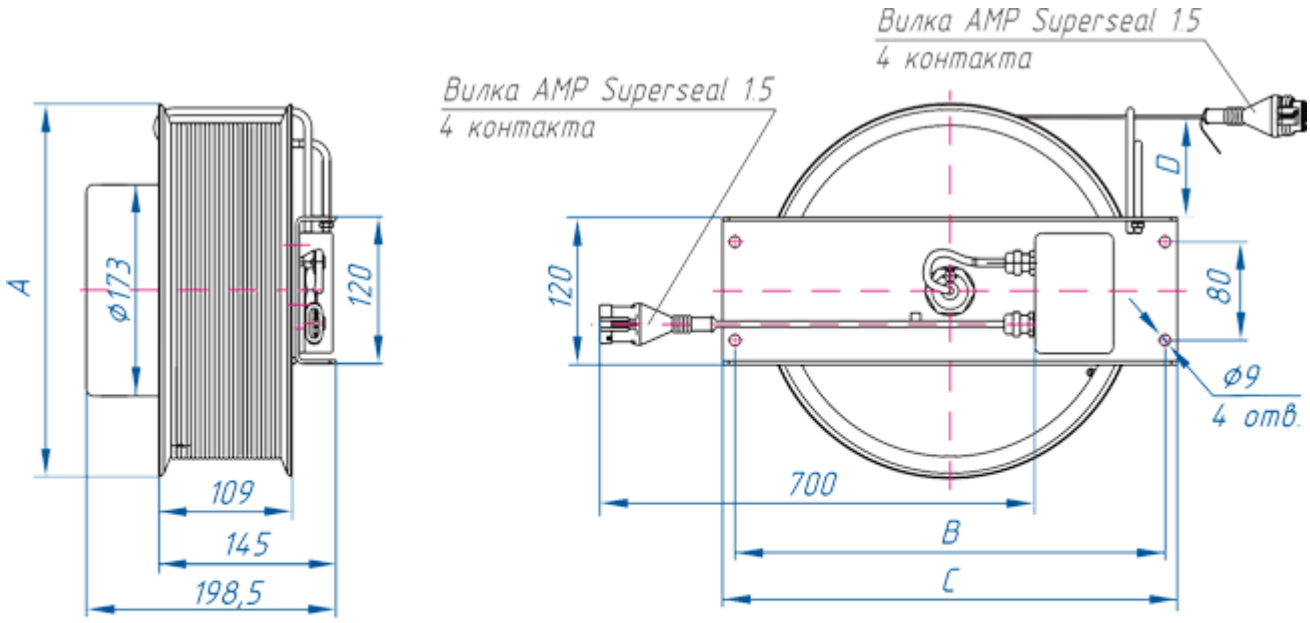
— измерение угла наклона относительно гравитационной нормали.

Таблица 6 - Основные технические данные

Параметр	Значение
Диапазон измерения, м	от 0 до 50
Погрешность измерения перемещения, м, не более	±0,1
Диапазон измерения угла наклона, град.	от 0 до 110
Погрешность измерения угла наклона, град., не более	±0,5
Тип выходного сигнала	цифровой последовательный на основе ISO 9141
Электрическое соединение	AMP Superseal 1,5
Напряжение питания, В	от 10 до 32
Потребляемый ток, А, не более	0,02
Максимально допустимый ток нагрузки, А	4
Диапазон температур, °С рабочих предельных	 от минус 40 до +55 от минус 50 до +65

Продолжение таблицы6

Степень защиты от внешних воздействующих факторов по ГОСТ 14254-96	IP56
Допустимые вибрационные нагрузки, не более	
максимальное ускорение, м/с2	50
в диапазоне частот, Гц	от 50 до 250
Допустимые ударные нагрузки, м/с2, не более	100
Габаритные размеры, мм, не более	305 × 370 × 199
Масса, кг, не более	16



Исполнение	A, мм	B, мм	C, мм	D, мм	Измерение длины	Измерение угла наклона
ДДС15	305	350	370	89	0–24 м	0–110°
ДДС30	385	426	466	127	0–32 м	0–110°
ДДС50	455	446	486	156	0–50 м	0–110°

Рисунок 5 - Датчики перемещения серии ДДС

Датчик азимута ДУА360.13

Основные технические данные содержатся в таблице7. Сборочный чертеж датчика представлен на рисунке6.

Основные особенности

— установка на токосъемное устройство крана;

— цифровой интерфейс передачи данных;

— наличие дискретного входа.

Датчик предназначен для измерения угла поворота валов. Как правило, используется в системах безопасности строительно-дорожных машин для измерения угла поворота крановой (бурильной, выдвижной и т.д.) установки относительно неподвижного шасси.

Таблица 7 - Основные технические данные

Параметр	Значение
Диапазон измерения угла поворота, град.	от 0 до 360
Погрешность измерения, град., не более	1
Тип выходного сигнала	цифровой последовательный на основе ISO 9141
Электрическое соединение	вилка AMP Superseal 1.5 (4 контакта)
Напряжение питания, В	от 8 до 32

Продолжение таблицы7

Параметр	Значение
Диапазон температур, °С рабочих предельных	 от минус 40 до +65 от минус 60 до +80
Степень защиты от внешних воздействующих факторов по ГОСТ 14254-96	IP56
Допустимые вибрационные нагрузки, не более	
максимальное ускорение, м/с^2	50
в диапазоне частот, Гц	от 50 до 200
Допустимые ударные нагрузки, м/с^2, не более	100
Габаритные размеры, мм, не более	118 × 96 × 89
Масса, кг, не более	0,9

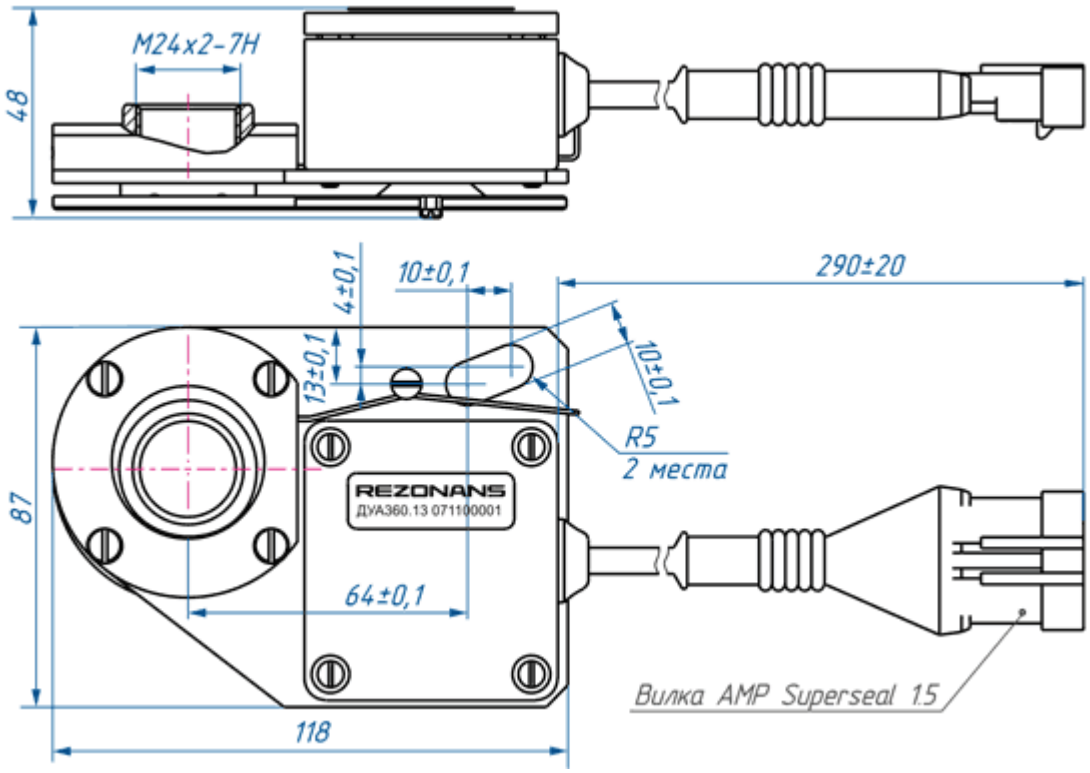


Рисунок 6 - Датчик азимута

Датчики усилия серии ТРС

Основные технические данные содержатся в таблице 8. Сборочный чертеж датчика представлен на рисунке 7.

Основные особенности

— измерение усилия до 10000 кгс;

— высокая точность и стабильность показаний, достигаемая за счет встроенной калибровки и термокомпенсации;

— цифровой интерфейс передачи данных;

— наличие дискретного входа.

Предназначены для определения усилия в составе приборов безопасности и систем управления грузоподъемной техники. Возможно применение в различных системах автоматического контроля, защиты и управления промышленного назначения, весовых системах и на строительно-дорожной технике.

Используются для измерения усилия в системах защиты и управления и приборах безопасности грузоподъемных машин.

Таблица 8 - Основные технические данные

Параметр	Значение
Диапазон измерения усилия, кгс	от 0 до 10000
Погрешность измерения, не более	1%
Вид деформации чувствительного элемента	растяжение
Допустимая перегрузка, не более	
уровень перегрузки, %	150
длительность перегрузки, мин.	2
Напряжение питания, В	от 10 до 32
Диапазон температур, °С рабочих хранения	от минус 40 до +55 от минус 50 до +65
Степень защиты от внешних воздействующих факторов по ГОСТ 14254-96	IP67
Допустимые вибрационные нагрузки, не более	
максимальное ускорение, м/с^2	50
в диапазоне частот, Гц	от 50 до 250

Продолжение таблицы 8

Параметр	Значение
Допустимые ударные нагрузки, м/с^2, не более	150
Масса, кг, не более	
до 1000 кгс	1,5
от 1000 до 5000 кгс	3,2
от 5000 до 10000 кгс	5,0

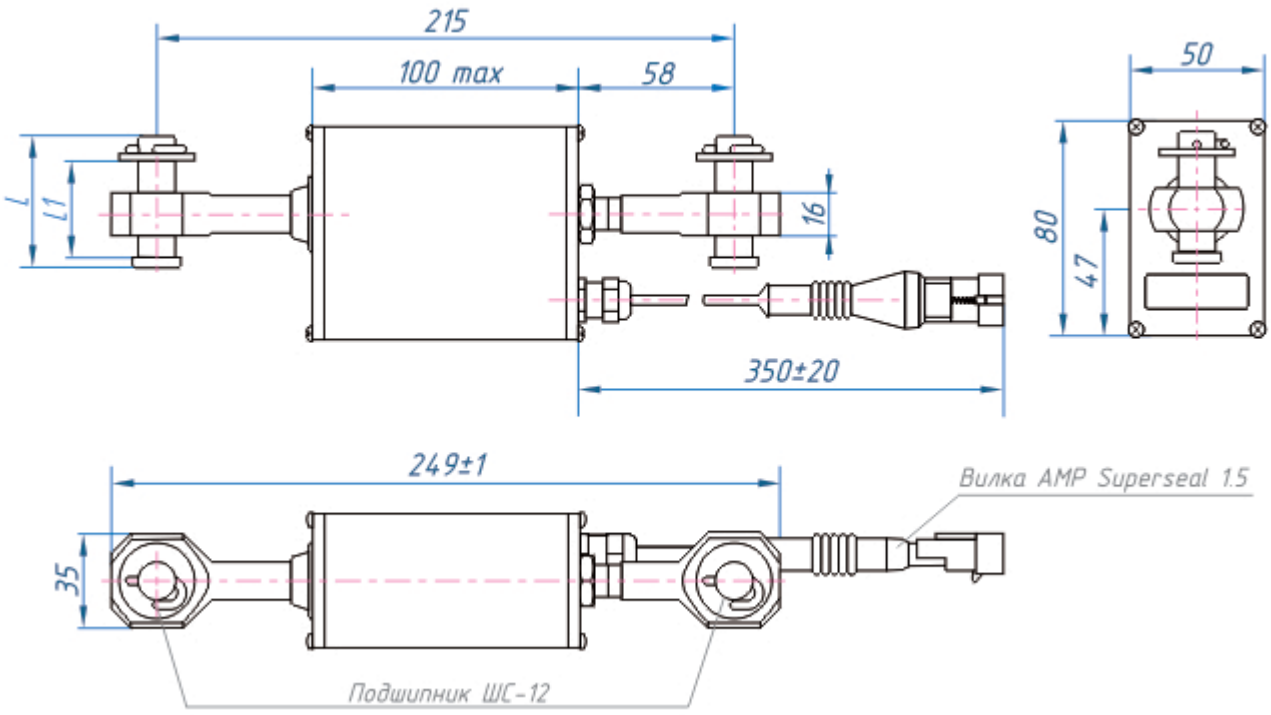


Рисунок 7- Габаритные и присоединительные размеры

Датчики угла наклона ДУГ50

Основные технические данные содержатся в таблице 9. Сборочный

чертеж датчика представлен на рисунке 8.

Основные особенности

— модификации для измерения углов наклона относительно одной или двух осей;
— высокая стойкость к воздействию вибраций, ударов и температур;
— аналоговые и цифровые интерфейсы передачи данных;
— герметичное исполнение.

Предназначены для измерения углов наклона рабочего оборудования строительно-дорожных машин относительно гравитационной вертикали.

Применяются в составе систем защиты и управления для измерения угла наклона стрелы грузоподъемного крана, продольного и поперечного наклона базовой платформы кранов и автолестниц, а также для контроля вертикальности рабочей мачты бурильно-крановых машин и т.п.

Таблица 9 - Основные технические данные

Параметр	Значение
Количество осей измерения угла наклона	1 или 2
Диапазон измерения угла наклона по каждой оси	от 0 до 110°
Погрешность измерения угла наклона, не более	±0,3°

Продолжение таблицы 9

Параметр	Значение
Диапазон температур: рабочих предельных	от минус 40 до +55 °С от минус 50 до +65 °С
Напряжение питания	от 10 до 32 В
Степень защиты от внешних воздействующих факторов по ГОСТ 14254-96	IP67
Допустимые вибрационные нагрузки:	
максимальное ускорение	50 м/с^2
в диапазоне частот	от 50 до 250 Гц
Допустимые ударные нагрузки	100 м/с^2
Габаритные размеры	73 × 40 × 25 мм
Масса	0,12 кг

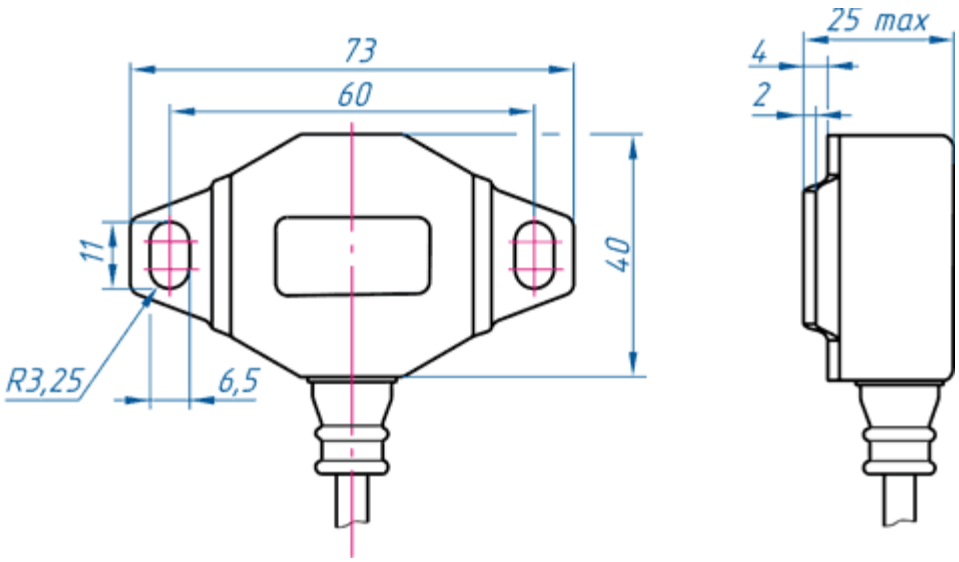


Рисунок 8 - Габаритные и присоединительные размеры

Преобразователи скорости ветра серии МС1

Основные технические данные содержатся в таблице 10. Сборочный чертеж датчика представлен на рисунке 9.

Основные особенности

— крепление на неподвижное основание;

— корпус и крыльчатка из армамида;

— четырехсекундное усреднение результатов измерения;

— модификации с цифровым или импульсным интерфейсом.

Предназначен для измерения скорости ветрового потока.

Применяется в составе приборов безопасности грузоподъемных кранов, автолестниц и автогидроподъемников для измерения скорости ветра и определения опасных ветровых порывов.

Датчик модификации МС1-И13 по присоединительным размерам и электрическим сигналам взаимозаменяем с датчиком скорости ветра ДСВ-2 анемометра АСЦ-3.

Таблица 10 - Основные технические данные

Параметр	Значение
Диапазон измерения, м/с	от 1,5 до 32
Погрешность измерения, м/с, не более V — измеряемое значение скорости ветра	± (1+0,1 V)
Выходной сигнал	
модификации МС1-Р11, МС1-Р13	RBus
модификации МС1-И11, МС1-И13	импульсный (0 – 32 Гц)
Электрическое соединение	
модификации МС1-Р11, МС1-И11	вилка AMP Superseal 1,5 (4 контакта)
модификации МС1-Р13, МС1-И13	разъем 2РМ14БПН4Ш1В1
Напряжение питания, В	от 10 до 32
Диапазон температур, °С рабочих предельных	 от минус 40 до +55 от минус 50 до +65
Степень защиты от внешних воздействующих факторов по ГОСТ 14254-96	IP55
Допустимые вибрационные нагрузки, не более	

Продолжение таблицы 10

Параметр	Значение
максимальное ускорение, м/с^2	50
в диапазоне частот, Гц	от 50 до 250
Допустимые ударные нагрузки, м/с^2, не более	100
Габаритные размеры (без крепления), мм, не более	161 × 64
Масса датчика, кг, не более	0,6

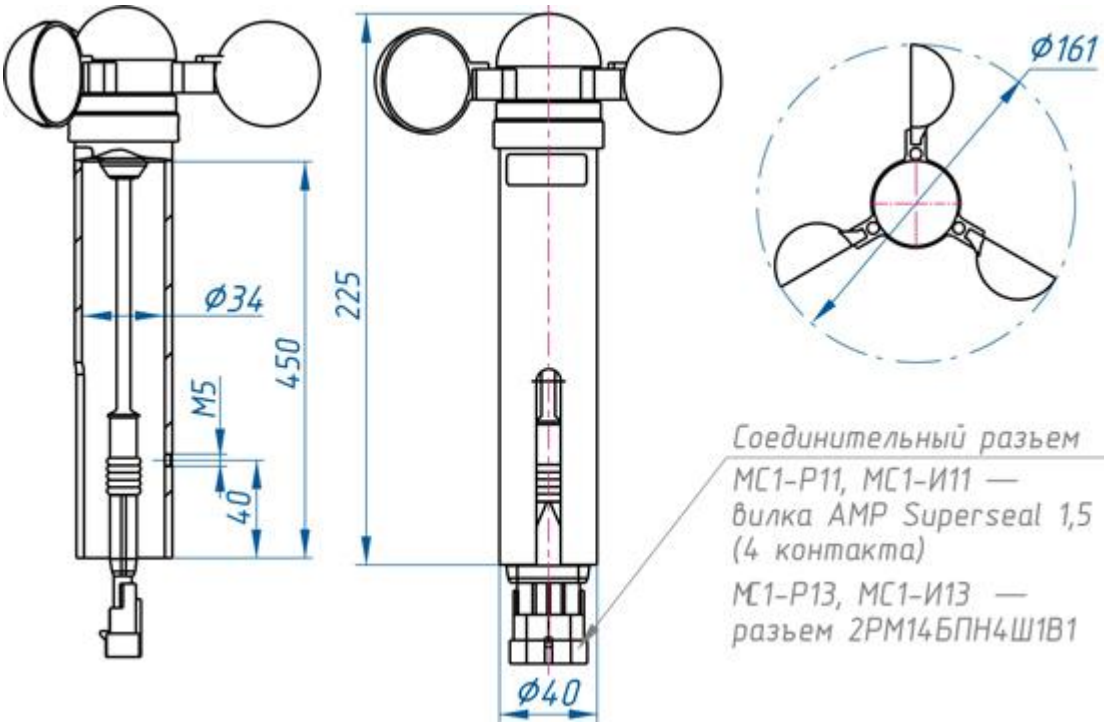


Рисунок 9- Габаритные и присоединительные размеры

Выводы главы:

В данной главе были рассмотрены аналоги блока индикации, а также аппаратные решения, имеющие важное значение для функционирования системы безопасности.

2 Разработка алгоритмов и программного обеспечения универсального блока индикации

В данной главе рассматривается реализация следующих задач:

— разработка элементов управления дисплеем блока индикации БИ04.70;
— разработка алгоритмов для реализации необходимых функций (Приложение Г);
— описание полученных алгоритмов для блока индикации БИ04.70 на языке С (приложения Б и В);

2.1 Требования к разработке элементов управления дисплеем блока индикации БИ04.70

Лицевая панель блока индикации изображена на рисунке 10. Назначение кнопок и элементов индикации содержится в таблице 11.

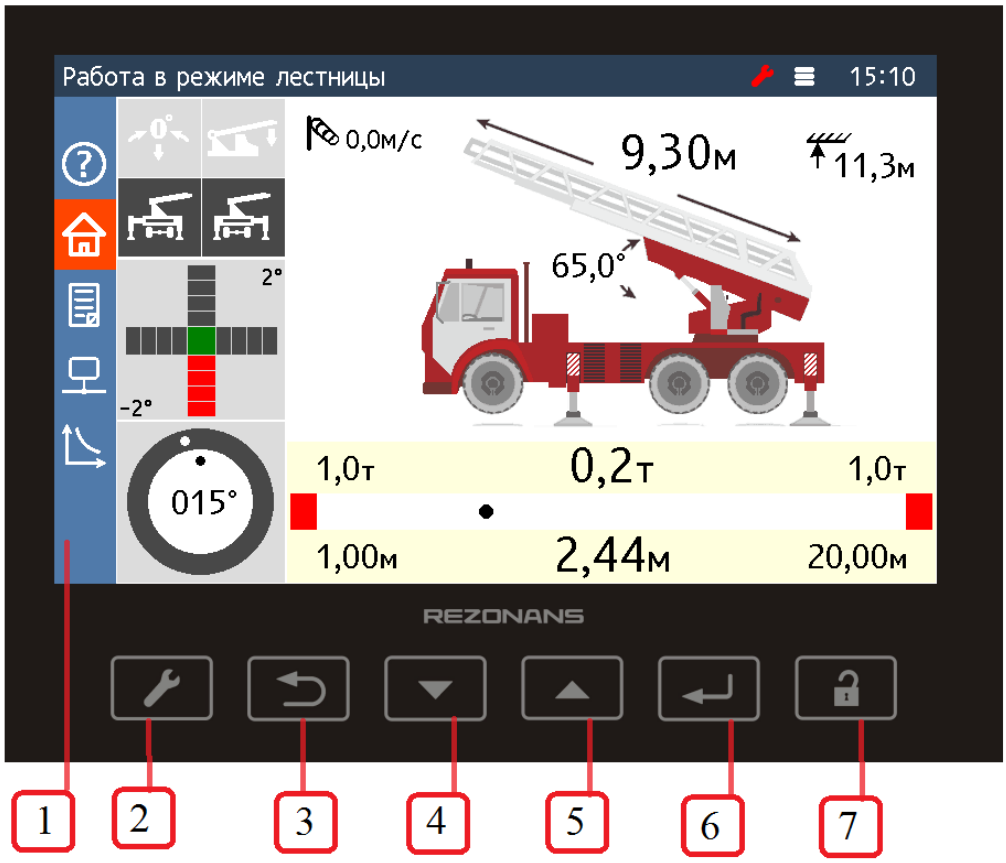


Рисунок 10— Лицевая панель блока индикации БИ04.70

Таблица 11— Назначение кнопок и элементов индикации

Номер элемента	Элемент и его функции
1	Цветной жидкокристаллический дисплей.
2	По кнопке «Меню» должны выполняться следующие действия : — переход к вводу/снятию режима перевода лестницы в транспортное положение; — переход к считыванию регистратора параметров; — переход к установке параметров блока индикации (уровень подсветки дисплея, уровень громкости звукового сигнала, язык интерфейса); — переход к меню настройки (когда включен режим настройки); — выходить из меню.

Продолжение таблицы 11

Номер элемента	Элемент и его функции
3	Кнопка «Возврат» должна осуществлять: — переход на основной экран; — переход на предыдущий уровень меню.
4	Кнопка «Вниз» должна осуществлять: — переход к следующему экрану; — перевод курсора на нижнюю строку меню; — уменьшение значения изменяемого параметра.
5	Кнопка «Вверх» должна осуществлять: — переход к предыдущему экрану; — перевод курсора на верхнюю строку меню; — увеличение значения изменяемого параметра.
6	Кнопка «Ввод» должна осуществлять:: — выбор пункта меню; — ввод измененного значения параметра.
7	По кнопке «Разблокировка» должно происходить снятие блокировок определенных механизмов автолесницы в зависимости от сработавшего ограничения.

Графический интерфейс блока индикации БИ04.70 должен состоять из трех частей (см.рисунок 11):

— статусная строка, расположенная в верхней части дисплея;

— панель с иконками экранов, расположенная в боковой части дисплея;

— текущий экран с параметрами, занимающий основную часть дисплея.

Назначение элементов интерфейса приведено в таблице 12.

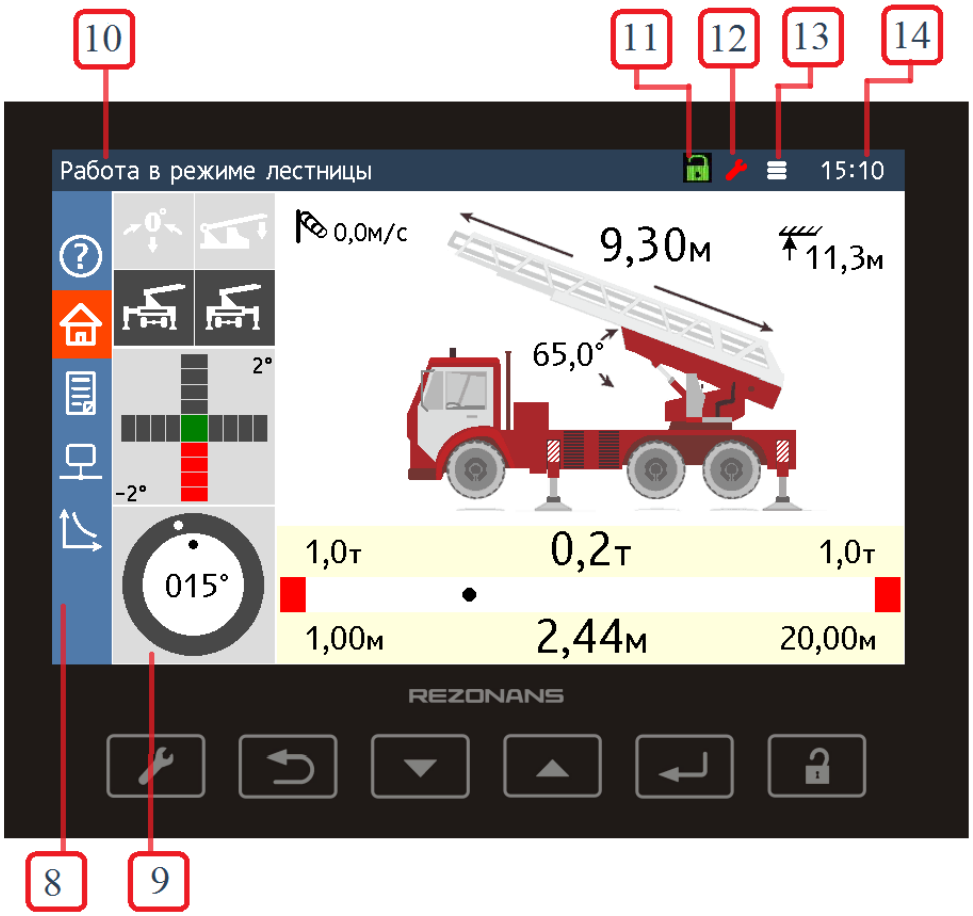


Рисунок 11— Графический интерфейс блока индикации БИ04.70

Таблица 12— Элементы графического интерфейса

Номер элемента	Элемент и его функции
8	Панель с иконками экранов.
9	Область активного экрана.
10	Статусная строка с информационными сообщениями.
11	Индикатор возможности снятия блокировки механизмов автолестницы.
12	Индикатор включения режима настройки.
13	Индикатор уровня яркости подсветки дисплея.
14	Текущее время.

После тестирования прибор безопасности должен переходить к основному экрану (см. рисунок 12). Описание элементов экрана приведено в таблице 13. На данном экране должны отображаться основные параметры автолесницы: текущий угол подъема лестницы (α, град), текущая длина лестничного марша (L, м), текущий вылет (Rт, м), максимально допустимый вылет (Rм, м), текущая высота подъема (H, м), текущий угол поворота платформы автолестницы (ω, град), текущая скорость ветра (V, м/с).

Для переключения между экранами используются кнопки 4 и 5.

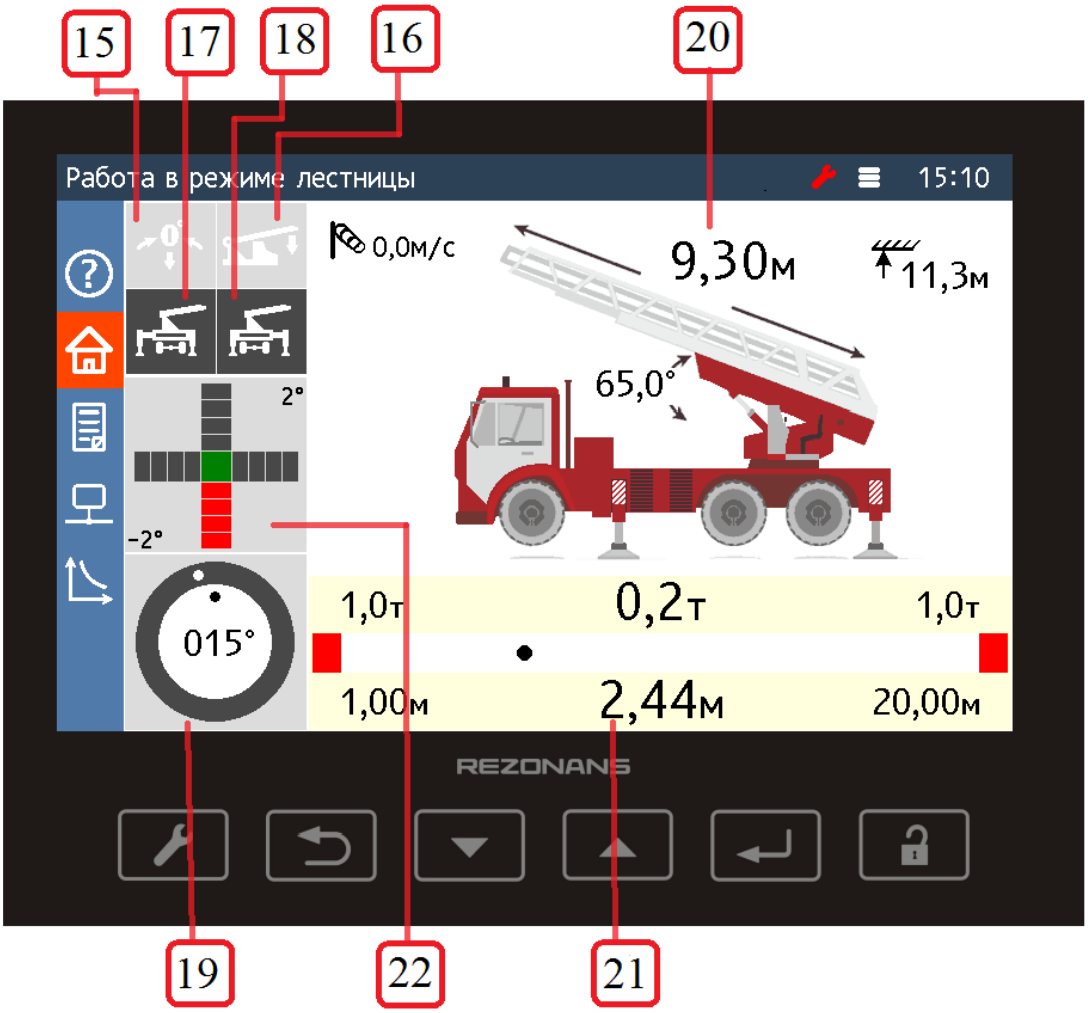


Рисунок 12— Основной экран

Таблица 13— Параметры основного экрана

Номер элемента	Элемент и его функции
15	Индикатор совмещения осей.
16	Индикатор кранового режима.
17	Индикатор выдвижения левых опор.
18	Индикатор выдвижения правых опор.
19	Индикатор угла поворота платформы автолестницы.
20	Область основных параметров.
21	Индикатор рабочей зоны.
22	Индикатор крена платформы

Экран информационных сообщений (см. рисунок 13) в табличной форме должен отображать все информационные сообщения, выводимые блоком индикации в текущий момент. На экране регистратора параметров(см. Рисунок 14) кроме моточасов должна приводиться сводная информация об автолестнице и приборе безопасности.

Рисунок 13 – Экран информационных сообщений

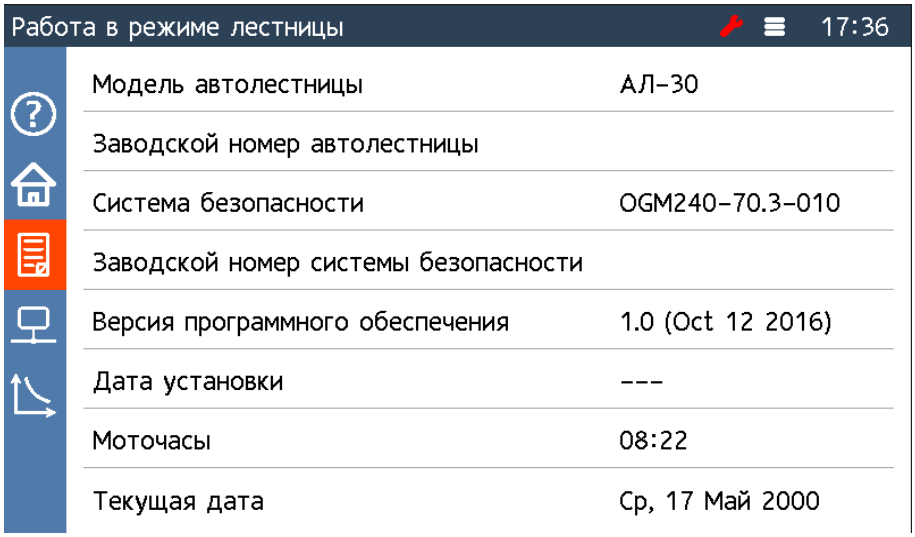


Рисунок 14— Экран регистратора параметров

Для диагностики подключенных датчиков в ОГМ240 используется экран приведенный на рисунке 15.

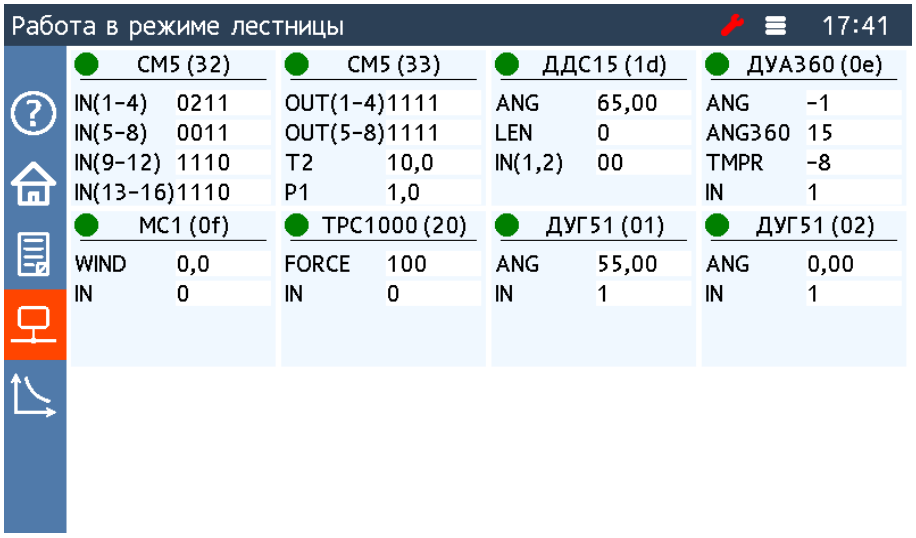


Рисунок 15— Экран информации с датчиков

Для отслеживания положения стрелы следует использовать экран, приведенный на рисунке16. Описание элементов экрана приведено в таблице 14.

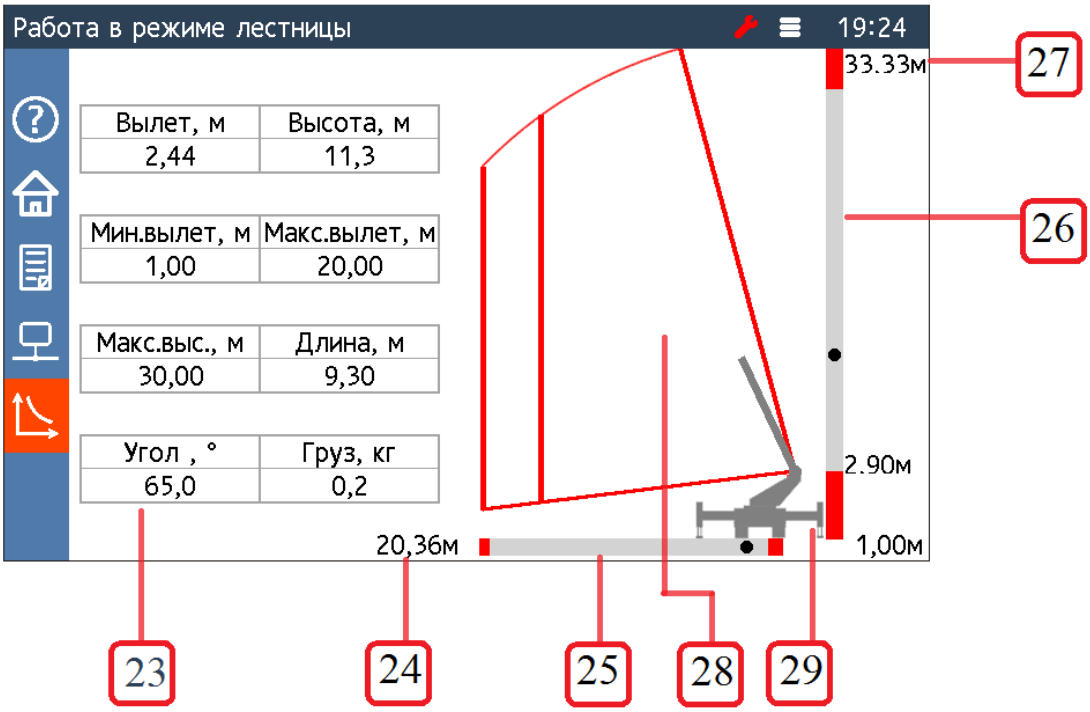


Рисунок 16 — Экран отображения положения стрелы

Таблица 14— Параметры экрана с зоной работы

Номер элемента	Элемент и его функции
23	Таблица с данными
24	Расчетный вылет для границы зоны по вылету в зависимости от длины стрелы.
25	Индикатор рабочей зоны по вылету.
26	Индикатор рабочей зоны по высоте.
27	Расчетная высота для границы зоны по высоте в зависимости от длины стрелы.
28	Изображение рабочей зоны.
29	Схематическое изображение машины со стрелой.

Установка параметров блока индикации, а также настройка ОГМ240 должна осуществляться в меню. Для перехода к меню необходимо нажать

кнопку 2 (см. рисунок 17). Для навигации по меню используются кнопки 4, 5 и 6. Для выхода из меню необходимо нажать кнопку 2.

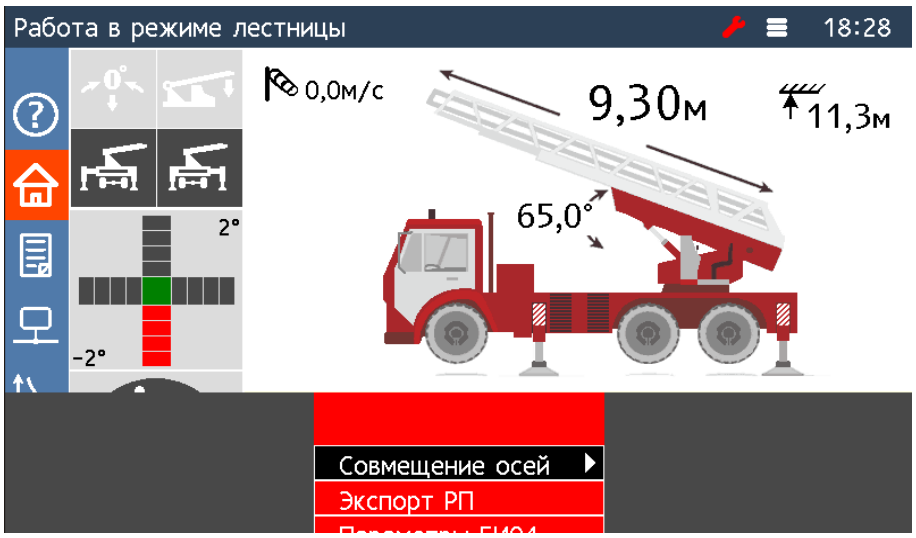


Рисунок17 — Меню блока индикации

2.2 Функции прибора безопасности

Ограничение рабочих движений автолестницы

Ограничитель рабочих движений лестницы автоматически формирует сигнал блокировки механизмов управления автолестницы при выходе за допустимые параметры работы. При возврате лестницы в разрешенную зону работы сигнал блокировки автоматически снимается.

Ограничения, предназначенные для предотвращения повреждений механизмов лестницы:

— ограничение превышения максимального вылета;

— ограничение превышения максимальной высоты;

— ограничение максимального выдвижения лестничного комплекта;

— ограничение работы со стороны невыдвинутых опор;

— ограничение работы в зоне кабины.

При срабатывании одного из ограничений формируется сигнал блокировки механизмов лестницы, при этом зажигается красное световое табло,

включается прерывистый звуковой сигнал и на дисплей выводится информационное сообщение. Снятие блокировки происходит автоматически после выхода из зоны ограничения.

При приближении к границе ограничения, прибор безопасности информирует оператора кратковременным звуковым сигналом. При этом включается желтое световое табло, формируется сигнал ограничения скорости движения автолестницы и на дисплей выводится информационное сообщение.

Прибор безопасности поддерживает работу с тремя максимально возможными вылетами: при работе с грузом больше 180 килограмм, при работе с грузом меньше 180 и при работе со стороны невыдвинутых опор. Величина всех трех вылетов определяется производителем автолестницы и задается в процессе настройки прибора безопасности. Выбор текущего максимального вылета производится автоматически в зависимости от состояния входов прибора безопасности.

В случае если при входе в зону невыдвинутых опор вылет лестницы окажется больше вылета, при котором формируется сигнал ограничения скорости движения или сигнал блокировки движения, то при приближении к зоне невыдвинутых опор так же будет сформирован сигнал ограничения скорости работы, включается желтое световое табло, на дисплей выводится информационное сообщение и включается кратковременный звуковой сигнал.

При входе в зону невыдвинутых опор путем поворота слева (или справа) формируется сигнал блокировки поворота в соответствующую сторону, зажигается красное световое табло, включается прерывистый звуковой сигнал. Снятие блокировки происходит автоматически после выхода из зоны невыдвинутых опор.

При опускании лестницы в зону кабины с углом поворота, соответствующим укладке лестницы на стойку, блокировка опускания не включается, но при этом включается блокировка поворотов вправо и влево. Для разрешения поворотов в зоне кабины для более точной укладки лестницы на стойку необходимо воспользоваться режимом перевода лестницы в транспортное положение.

Ограничение грузоподъемности

При перегрузке прибор безопасности формирует сигнал блокировки механизмов лестницы, на блоке индикации загорается красное световое табло и включается прерывистый звуковой сигнал. Выход из блокировки происходит после снятия перегрузки.

Максимальный груз при работе в крановом режиме — 1 тонна.

Перевод лестницы в транспортное положение

Для перевода лестницы в транспортное положение необходимо:

— нажать кнопку 2 для входа в меню;
— кнопками 4, 5 выбрать пункт «Совмещение осей», нажать кнопку 6;
— нажать кнопку 6, при этом произойдет переход в режим перевода лестницы в транспортное положение(см. рисунок 18).

При приближении к положению совмещения осей формируется сигнал ограничения скорости работы. При совмещении осей базового шасси и лестницы формируются сигналы запрета поворотов. При опускании лестницы в зону укладки будут сформированы сигналы разрешения поворотов вправо и влево для возможности более точной укладки лестницы на транспортную стойку. При этом в случае, если лестница будет повернута на угол больше, чем угол укладки на стойку, будет сформирован сигнал запрета поворота в соответствующую сторону. Если выход из зоны укладки

лестницы на стойку превысит 1 град, то будут сформированы оба сигнала запрета поворотов. В этом случае необходимо поднять лестницу над зоной кабины и повторить процедуру укладки лестницы на стойку.

Для выхода из режима перевода лестницы в транспортное положение необходимо:

— нажать кнопку 2 для входа в меню;

— кнопками 4, 5 выбрать пункт «Совмещение осей», нажать кнопку 6;

— повторно нажать кнопку 6.

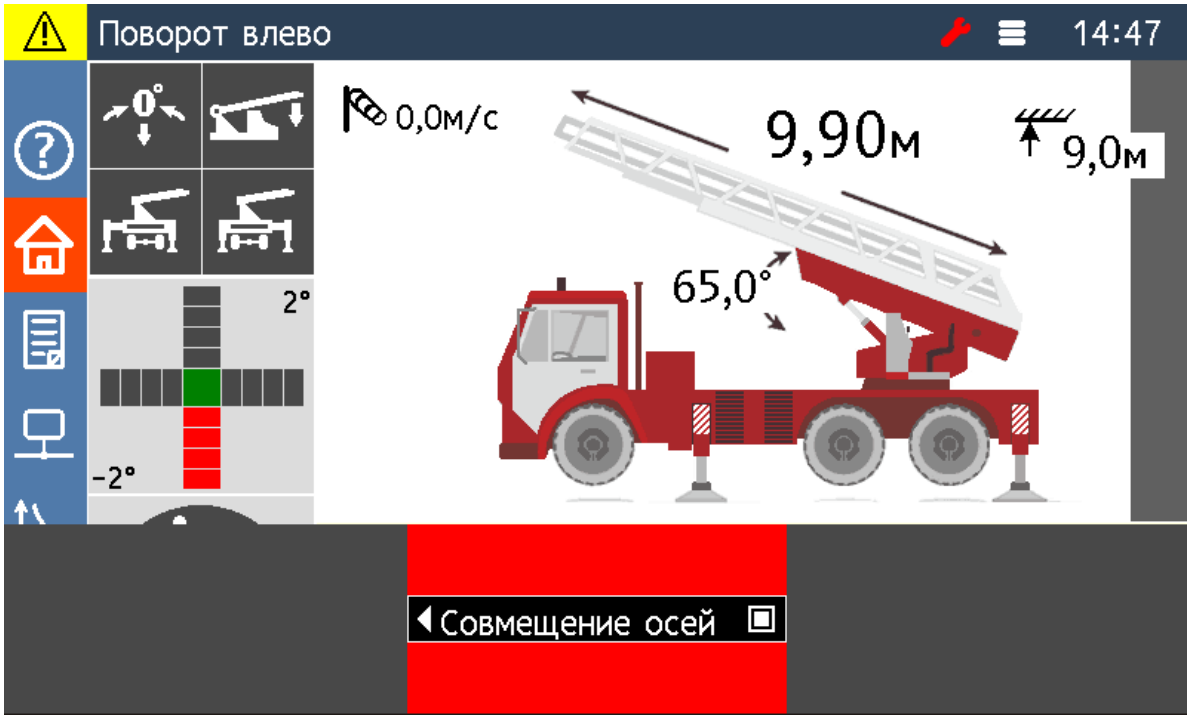


Рисунок 18 — Установка режима «Совмещение осей»

Определение скорости ветра

При фиксировании скорости ветра более максимально допустимой на дисплей выводится сообщение о предельной скорости ветра, на блоке индикации загорается красное световое табло и включается прерывистый звуковой сигнал. Выход из блокировки происходит при снижении скорости ветра.

Регистратор параметров

Регистратор параметров состоит из двух областей памяти, предназначенных для хранения:

— оперативной информации;

— долговременной информации;

Оперативная информация состоит из набора записей. Одна запись включает в себя:

— дату и время записи;

— угол наклона лестницы;

— угол поворота платформы автолестницы;

— вылет;

— высоту;

— длину лестничного марша;

— информацию о сработавших ограничениях.

Записи оперативной информации производятся с периодом 1 с при наличии блокировок механизмов автолестницы и с периодом 25 с при отсутствии блокировок.

Долговременная информация включает в себя:

— общую наработку автолестницы в моточасах;

— номер автолестницы и номер прибора безопасности;

— дату установки прибора безопасности на автолестницу.

Обработка и распечатка данных регистратора параметров осуществляется на персональном компьютере (ПК) под управлением операционной системы Windows с помощью программы LogSystem.

Передача данных на ПК производится с помощью USB Flash носителя.

Для считывания РП необходимо:

— нажав кнопку 2 перейти в главное меню;

— нажатием кнопки 4 выбрать пункт меню «Экспорт РП», нажать кнопку 6;

— вставить USB Flash носитель в блок индикации, нажать кнопку 6(см. рисунок 19);

— дождаться появления на дисплее сообщения «данные записаны», извлечь носитель из блока индикации;

— скопировать файл с носителя в папку на ПК;

— в папке с записанным файлом запустить LogConverter, результатом работы программы станут файлы с расширением lgs.

Обработка и распечатка данных регистратора производится согласно руководству пользователя программы Rezonans LogSystem.

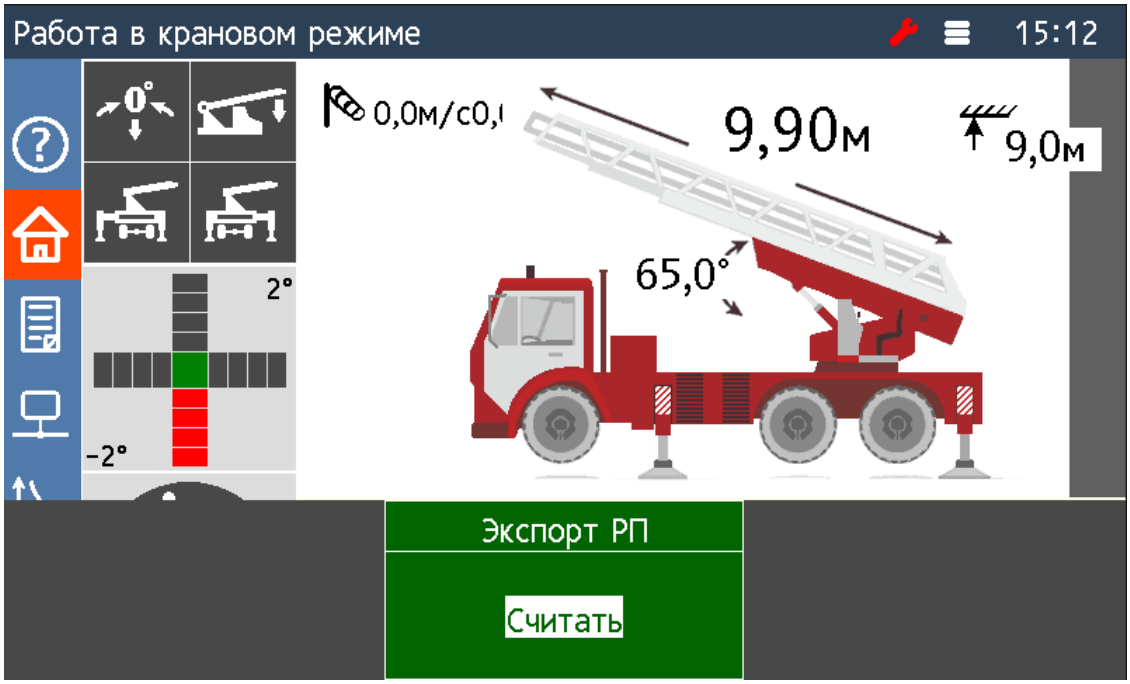


Рисунок 19— Считывание параметров

Тестирование составных частей

В ходе работы ОГМ240 производит автоматическое тестирование составных частей, при обнаружении неисправности все механизмы

автолестницы блокируются, в статусной строке дисплея выводится соответствующее информационное сообщение.

2.3 Испытания, необходимые для проверки работоспособности системы безопасности

Предварительная проверка функций блока индикации производилась путём симуляции данных с датчиков с помощью следующих средств: адаптер связи и программа VirtDat. Схема проверки изображена в приложении Д.

Работоспособность системы безопасности, в состав которой входил блок индикации БИ04.70 с разработанным программным обеспечением, оценивалась на основании испытаний на пожарной машине АЛ-30, проведённых ОАО "Пожтехника" г. Торжок.

План проведения испытаний для пожарной машинs проводился согласно основным техническим требованиям и методам испытаний, изложенных в нормами пожарной безопасности 188-2000.

Выводы главы:

В данной главе были подробно описаны основные требования к программному обеспечению, функции, которые должен выполнять блок индикации. В завершение главы был приложен протокол испытаний для проверки системы безопасности автолесницы.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы была достигнута цель работы, а именно было написано программное обеспечение для универсального блока индикации для техники специального назначения.

В рамках выполнения поставленной задачи были синтезированы алгоритмы, реализующие функции прибора безопасности. Программное обеспечение было реализовано для пожарной машины АЛ-30, определены способы расчёта рабочих параметров системы, выбраны необходимые виджеты для наглядного демонстрирования характеристик системы оператору.

В программном обеспечении реализованы функции безопасности, основная задача которых — помочь оператору в управлении механизмами лестницы, не допустить повреждения или опрокидывания машины. Это достигается путем ограничений параметров или режимов работы автолестницы в следующих ситуациях:

— при достижении максимально допустимого вылета лестницы;
— при максимальном и минимальном выдвижении лестницы;
— при достижении максимального или минимального угла наклона лестницы;
— при работе в зоне кабины (для предотвращения повреждения кабины и транспортной стойки лестницы);
— при работе со стороны невыдвинутых опор;
— в случае превышения номинальной грузоподъемности;
— при срабатывании датчиков удара лестницы о препятствие (ограничитель лобового удара);
— при превышении допустимой скорости ветра;
— в других аварийных или потенциально-опасных ситуациях.

Реализовано отображение положения стрелы в рабочей зоне: написан виджет для демонстрации пространственного положения стрелы и отображение рабочей зоны для стрелы. Данная функцияпозволит более легко определять расстояние на котором следует расположить автолесницу, так как оператор может определять максимально возможное выдвижение лестницы без опасений опрокинуть машину.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Система управления СБУК322.07-000-001: Руководство по настройке / Кадыров А.К. –05.07.2016 – 74 с.

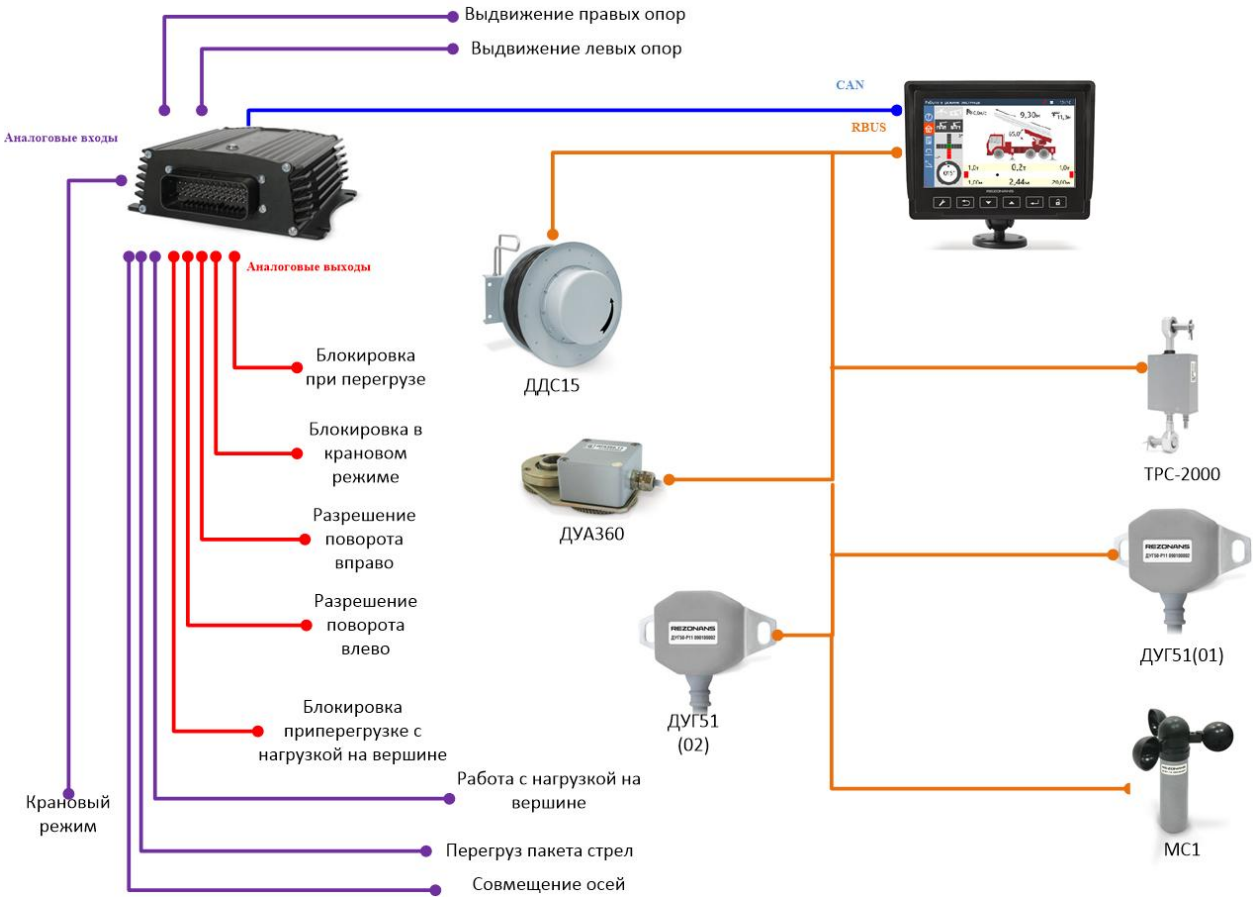
2. STM32F205xx STM32F207xx ARM-based 32-bit MCU, 150DMIPs, up to 1 MB Flash/128+4KB RAM, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 15 comm. interfaces & camera– Datasheet – 177 с.

3. STM32F205xx, STM32F207xx, STM32F215xx and STM32F217xx advanced ARM-based 32-bit MCUs: Reference Manual – 1361 с.

4. Нормы пожарной безопасности. Автолестницы пожарные. Основные технические требования. Методы испытаний. / В.В. Самохвалов, А.М. Петров, Л.С. Черткова - Москва: 2000 – 77 с.

ПРИОЛОЖЕНИЕ А

Схема электрическая соединений для системы безопасности пожарной машины АЛ-30



ПРИЛОЖЕНИЕ Б

Код страницы отображения положения стрелы в зоне работы

```
#include "global_includes.h"

#include "gui/gui.h"

#include "grlib/grlib.h"

#include "grlib/widget.h"

#include "gui_conf.h"

#include "limiter_pbl/limiter_pbl.h"

#include "gui/widgets/Highwidget.h"

#include "fstore/fstore_pbl_config.h"

uint8_t Flag_AL_30=0;

void Zone_Vilet_draw(tWidget *pWidget, tContext *pContext);

extern const unsigned char Zone4[];

extern const unsigned char AL_30_Grey[];

extern tArrowWidget Arrow_position;

__weak Zone_Page_struct Zone_Page;

//static void ZonePageFunc(tWidget *pWidget, tContext *pContext);

void Zero_function(tWidget *pWidget, tContext *pContext);

long Zero_function2(tWidget *pWidget, unsigned long ulMsg, unsigned long ulParam1, unsigned
    long ulParam2);

void Zero_function3(tWidget *pWidget, tContext *pContext);

void Zero_function4(tWidget *pWidget, tContext *pContext);


void ZonePageFunc(tWidget *pWidget, tContext *pContext);

// STR_GRLANGS          GrLangDe            GrLangEnUS          GrLangEsSP

// STR_Zone             "Зонаработы"        "Work zone"         "Work zone"

/*Canvas(g_sZonePage, 0, 0, &Arrow_position, MAIN_DISPLAY,
```

```
        NAVBAR_WIDTH, HEAD_HEIGHT, PAGE_WIDTH, PAGE_HEIGHT,
        CANVAS_STYLE_FILL|CANVAS_STYLE_APP_DRAWN,
        ClrWhite, 0, 0, 0, 0, 0, &ZonePageFunc);

ArrowActive(Arrow_position, &g_sZonePage, 0, 0, MAIN_DISPLAY,
        NAVBAR_WIDTH + 36, HEAD_HEIGHT, 650, 425,
        (tFont *)g_pucFont20pt, ClrGray, ClrWhite, ClrWhite, ClrWhite, ArrowProc,
      STR_Zone, 2, f2long(0), f2long(0), -7, 90, 90);

Canvas(g_ZoneHighCanvas, 0, 0, 0,
     MAIN_DISPLAY, NAVBAR_WIDTH +655, HEAD_HEIGHT, HIGHWIDGET_WIDTH,
     HIGHWIDGET_HEIGHT,
     CANVAS_STYLE_OUTLINE | CANVAS_STYLE_FILL | CANVAS_STYLE_APP_DRAWN,
     ClrWhite, ClrWhite, 0, 0, 0, 0, Zone_High_draw);
     //0xFFFFDD, ClrWhite, 0, 0, 0, 0, Zone_High_draw);

Canvas(g_ZoneViletCanvas, 0, 0, 0,
     MAIN_DISPLAY, NAVBAR_WIDTH +36+626-IMAGE_WIDTH, IMAGE_HEIGHT +
     SHIFT_IMAGE_TOP, VILET_WIDTH, VILET_HEIGHT,
     CANVAS_STYLE_OUTLINE | CANVAS_STYLE_FILL | CANVAS_STYLE_APP_DRAWN,
     ClrWhite, ClrWhite, 0, 0, 0, 0, Zone_Vilet_draw);
     //0xFFFFDD, ClrWhite, 0, 0, 0, 0, Zone_Vilet_draw);*/

Canvas(g_sZonePage, 0, 0, &Arrow_position, MAIN_DISPLAY,
        NAVBAR_WIDTH, HEAD_HEIGHT, PAGE_WIDTH, PAGE_HEIGHT,
        CANVAS_STYLE_FILL|CANVAS_STYLE_APP_DRAWN,
        ClrWhite, 0, 0, 0, 0, 0, &Zero_function);

ArrowActive(Arrow_position, &g_sZonePage, 0, 0, MAIN_DISPLAY,
```

```
        NAVBAR_WIDTH + 36, HEAD_HEIGHT, 650, 425,
        (tFont *)g_pucFont20pt, ClrGray, ClrWhite, ClrWhite, ClrWhite, Zero_function2,
       //  stop_icon, STR_Zone, 2, f2long(0), f2long(0), -7, 90, 90);
      STR_Zone, 2, f2long(0), f2long(0), -7, 90, 90);
Canvas(g_ZoneHighCanvas, 0, 0, 0,
    MAIN_DISPLAY, NAVBAR_WIDTH +655, HEAD_HEIGHT, HIGHWIDGET_WIDTH,
    HIGHWIDGET_HEIGHT,
    CANVAS_STYLE_OUTLINE | CANVAS_STYLE_FILL | CANVAS_STYLE_APP_DRAWN,
    ClrWhite, ClrWhite, 0, 0, 0, 0, Zero_function3);
    //0xFFFFDD, ClrWhite, 0, 0, 0, 0, Zone_High_draw);

Canvas(g_ZoneViletCanvas, 0, 0, 0,
    MAIN_DISPLAY, NAVBAR_WIDTH +36+626-IMAGE_WIDTH, IMAGE_HEIGHT +
    SHIFT_IMAGE_TOP, VILET_WIDTH, VILET_HEIGHT,
    CANVAS_STYLE_OUTLINE | CANVAS_STYLE_FILL | CANVAS_STYLE_APP_DRAWN,
    ClrWhite, ClrWhite, 0, 0, 0, 0, Zero_function4);

void Zero_function(tWidget *pWidget, tContext *pContext)
{
 if(Zone_Page.Zero_function)
 {
  Zone_Page.Zero_function(pWidget, pContext);
 }
 return;
}
long Zero_function2(tWidget *pWidget, unsigned long ulMsg, unsigned long ulParam1, unsigned
    long ulParam2)
{
 long i;
 if(Zone_Page.Zero_function2) i=Zone_Page.Zero_function2(pWidget, ulMsg, ulParam1, ulParam2);
```

```
 return i;
}
void Zero_function3(tWidget *pWidget, tContext *pContext)
{
 if(Zone_Page.Zero_function3) Zone_Page.Zero_function3(pWidget, pContext);
 return;
}
void Zero_function4(tWidget *pWidget, tContext *pContext)
{
 if(Zone_Page.Zero_function4) Zone_Page.Zero_function4(pWidget, pContext);
 return;
}
#include "gui/lib/table.h"
static Ttable tbl;
const TZone_param (*pZone_param)[] = 0;
uint8_t Zone_param_num = 0;

static void update(tWidget *pWidget)
{
    tContext sContext;
    tRectangle rect;
    uint8_t col;
    uint8_t param_num, tbl_ndx;
    if (pZone_param == 0 || Zone_param_num == 0)
        return;
    GrContextInit(&sContext, pWidget->pDisplay);
    GrContextClipRegionSet(&sContext, &pWidget->sPosition);
    GrContextBackgroundSet(&sContext, ClrWhite);
```

```
// устанавливаемшрифттаблицы
GrContextFontSet(&sContext, g_psFont24pt);
// задаемразмерытаблицы
memcpy(&rect, &sContext.sClipRegion, sizeof(tRectangle));
      char str[25];
tRectangle rect_cell, rect_ctx;
tbl_ndx = 0;
param_num = Zone_param_num;
rect.sXMin += 9;
 rect.sXMax = rect.sXMin+(rect.sXMax-rect.sXMin)*17/40;
//rect.sXMax -= 36;
rect.sYMin = 50 + 36;
           tbl.rows = 2;
tbl.cols = 2;
tbl.border_size = 2;
tbl.border_inner_size = 1;
tbl.border_style = TABLE_BORDER_ALL;
tbl.ppos = &rect;
tbl.col_width_ue[0] = 0;
do {
           if (tbl_ndx)
                      rect.sYMin = rect.sYMax + 36;
           //rect.sYMax = rect.sYMin + 60-1;
       rect.sYMax = rect.sYMin + 60-1;
           // инициализация и отрисовка рамки таблицы
           table_init(&tbl);
           GrContextForegroundSet(&sContext, 0xA5A5A5);
           table_paint_border(&sContext);
```

```
        GrContextForegroundSet(&sContext, ClrBlack);

        // сохраняемгеометриюконтекста
        memcpy(&rect_ctx, &sContext.sClipRegion, sizeof(tRectangle));
        for (col = 0; col < tbl.cols && param_num; col++, param_num--) {

            // получаем координаты текущей ячейки
            //  и ограничиваем ими контекст
            table_get_cell(0, col, &rect_cell);
            GrContextClipRegionSet(&sContext, &rect_cell);

            // выводимназваниепараметра
            GrStringGet((*pZone_param)[tbl.cols*tbl_ndx + col].name_id, str, sizeof(str));
            StringDrawAligned(&sContext, str, ALIGN_CENTER);

            // выводимзначениепараметра
            table_get_cell(1, col, &rect_cell);
            GrContextClipRegionSet(&sContext, &rect_cell);
            param2str(str, sizeof(str), (*pZone_param)[tbl.cols*tbl_ndx + col].ptr,
(*pZone_param)[tbl.cols*tbl_ndx + col].type, dig_sep, 0);
            StringDrawAligned(&sContext, str, ALIGN_CENTER);
        }

        // восстанавливаемгеометриюконтекста
        memcpy(&sContext.sClipRegion, &rect_ctx, sizeof(tRectangle));

// пока есть параметры в списке и не больше шести таблиц
} while (param_num && ++tbl_ndx < 4);
```

```
}
static long
CustomMsgProc(tWidget *pWidget, unsigned long ulMsg, unsigned long ulParam1,
        unsigned long ulParam2)
{
    switch(ulMsg) {
    case WIDGET_MSG_VALUE:
            if (WidgetIsInTree(WIDGET_ROOT, pWidget))
                    update(pWidget);
            return (1);
    default:
            return CanvasMsgProc(pWidget, ulMsg, ulParam1, ulParam2);
    }
}
void ZonePageFunc(tWidget *pWidget, tContext *pContext)
{

    pWidget->pfnMsgProc = CustomMsgProc;
    update(pWidget);
}
const struct param_place Zone_Vilet_param_places[] = {
      { LOADINDICATOR__R,     170, PARAM_FLOAT_2, FONT_44, STR_m, ALIGN_CENTER }, // r
    { LOADINDICATOR__RMIN,  120, PARAM_FLOAT_2, FONT_24, STR_m, ALIGN_LEFT },   // rmin
    { LOADINDICATOR__RMAX,  120, PARAM_FLOAT_2, FONT_24, STR_m, ALIGN_RIGHT },  // rmax
    расчетное
    { LOADINDICATOR__RMAX2,  120, PARAM_FLOAT_2, FONT_24, STR_m, ALIGN_RIGHT },  //
    rmax полученное
};
unsigned long Zone_Vilet_values[sizeof(Zone_Vilet_param_places)/
```

```
                    sizeof(*Zone_Vilet_param_places)];
float value[2];
void Zone_Vilet_param_draw(tWidget *pWidget, int nparam)
{
    tContext sContext;
    tRectangle rect;
    const tFont *font;
    unsigned long bar_width;
     //int Red_zone_RMin; //длина красной зоны для минимального вылета
      //Границы виджета и его ширина
     rect.sXMin = pWidget->sPosition.sXMin;
    rect.sXMax = pWidget->sPosition.sXMax;
GrContextInit(&sContext, pWidget->pDisplay);
      //Вывод на экран максимального и минимального вылета
if ((Zone_Vilet_param_places[nparam].id !=
    LOADINDICATOR__R)&&(Zone_Vilet_param_places[nparam].id != LOADINDICATOR__RMAX2))
    //Значение высоты будет выводиться в таблице, а не на виджете высоты
{
    if (Zone_Vilet_param_places[nparam].font <= font_family_nums) font =
    font_family[Zone_Vilet_param_places[nparam].font];
    else font = font_family[Zone_Vilet_param_places[0].font];
    GrContextFontSet(&sContext, font);
     switch (Zone_Vilet_param_places[nparam].id) {
    case LOADINDICATOR__RMAX:
            rect.sXMin -= SHIFT_VILET_VALUE;
            rect.sXMax = rect.sXMin + Zone_Vilet_param_places[nparam].w;
            break;
    case LOADINDICATOR__RMIN:
            rect.sXMax += SHIFT_VILET_VALUE;
            rect.sXMin = rect.sXMax - Zone_Vilet_param_places[nparam].w;
```

```
            break;
    }
    switch (Zone_Vilet_param_places[nparam].id) {
    case LOADINDICATOR__RMIN:
    case LOADINDICATOR__RMAX:
            // с учётом CANVAS_STYLE_OUTLINE (контур в 1 пиксель вокруг виджета)
            rect.sYMin = pWidget->sPosition.sYMin-10;
            rect.sYMax = pWidget->sPosition.sYMin-10+(pWidget->sPosition.sYMax-pWidget-
>sPosition.sYMin+BAR_HEIGHT);
            break;
    }

    GrContextClipRegionSet(&sContext, &rect);
    char string[32];
    char sunit[5];
    if (Zone_Vilet_param_places[nparam].unit)
            GrStringGet(Zone_Vilet_param_places[nparam].unit, sunit, sizeof(sunit));
    else sunit[0] = '\0';
    // преобразуем значение параметра в строку
    param2str(string, sizeof(string), &Zone_Vilet_values[nparam],
    Zone_Vilet_param_places[nparam].type, dig_sep, sunit);

    // выводим строку по центру заданной области
    GrContextForegroundSet(&sContext, ClrBlack);
    GrContextBackgroundSet(&sContext, ClrWhite);
    StringDrawAligned(&sContext, string, Zone_Vilet_param_places[nparam].align);
}
    if (Zone_Vilet_param_places[nparam].id != LOADINDICATOR__R
    && Zone_Vilet_param_places[nparam].id != LOADINDICATOR__RMIN
```

```
    && Zone_Vilet_param_places[nparam].id != LOADINDICATOR__RMAX
&& Zone_Vilet_param_places[nparam].id != LOADINDICATOR__RMAX2)
            return;
/*-----------------------------------------------------------------------------------------
    Вычисдение координаты начала шкалы вылета в зависимости от значения поправки по
    вылету
    LengthR0 - расстояние в отсчетах между координатой начала стрелы и координатой начала
    шкалы
    KoordR0 - координата начала шкалы вылета
-----------------------------------------------------------------------------------------*/
union {
        float    fl;
        unsigned long dw;
    } f;
    long LengthR0, KoordR0;
    rect.sXMin = pWidget->sPosition.sXMin;
    rect.sXMax =pWidget->sPosition.sXMax;

    float RMax, RMin, R, RMax2;          //значение RMax, RMin, R
    f.dw = Zone_Vilet_values[2];
    RMax = f.fl;
    f.dw = Zone_Vilet_values[1];
    RMin = f.fl;
    f.dw = Zone_Vilet_values[0];
    R = f.fl;
    f.dw = Zone_Vilet_values[3];
    RMax2 = f.fl;
    // lim_config.r_axis - Rпопр
    LengthR0= (long)((pbl_config.r_axis/(RMax+pbl_config.r_axis))*(rect.sXMax-
    SHIFT_ARROW_LEFT-rect.sXMin));
```

```
KoordR0=rect.sXMin+(rect.sXMax-rect.sXMin-SHIFT_ARROW_LEFT-LengthR0);


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

Вычисдение точки положения точки на шкале вылета:

LengthR0 - расстояние в отсчетах между координатой начала шкалы и координатой точки

KoordR - координата положения точки

LengthRmin - длина красной зоны с (минимальный вылет)

LengthRmax - длина красной зоны с (максимальный вылет)

long LengthR, LengthRmin, LengthRmax;

LengthR=(long)((R/RMax)*(rect.sXMax-SHIFT_ARROW_LEFT-LengthR0-rect.sXMin));

//KoordR=KoordR0-LengthR;

LengthRmin=(long)((RMin/RMax)*(rect.sXMax-SHIFT_ARROW_LEFT-LengthR0-rect.sXMin));

if (LengthRmin<LIMIT_WIDTH) LengthRmin=LIMIT_WIDTH;

LengthRmax=(long)(((RMax-RMax2)/RMax)*(rect.sXMax-SHIFT_ARROW_LEFT-LengthR0-
rect.sXMin));

if (LengthRmax<LIMIT_WIDTH) LengthRmax=LIMIT_WIDTH;

    bar_width = KoordR0 - rect.sXMin + 1;  //длинашкалывылетавотсчетах

// задаем координаты обасти графического представления вылета

rect.sYMin = pWidget->sPosition.sYMin; //+ (pWidget->sPosition.sYMax - pWidget->sPosition.sYMin
+ 1 - BAR_HEIGHT) / 2;

rect.sYMax = rect.sYMin + BAR_HEIGHT - 1;

//Red_zone_RMin=ceil(Zone_Vilet_values[2]);

// modf(Zone_Vilet_values[2], &Red_zone_RMin);

//Red_zone_RMin=(int)(Zone_Vilet_values[2]);

// Red_zone_RMin=((float)(Zone_Vilet_values[1]/Zone_Vilet_values[2])*((pWidget-
>sPosition.sXMax - 4 - SHIFT_VILET0_LEFT)-(pWidget->sPosition.sXMin + 4)));
//вычисляемдлинукрасной зоныдля RMin

// ограничиваемконтекст

GrContextClipRegionSet(&sContext, &rect);

// рисуемлевуюграницу
```

```
 rect.sXMin = pWidget->sPosition.sXMin;
rect.sXMax = rect.sXMin + LengthRmax - 1;
GrContextForegroundSet(&sContext, ClrRed);
GrRectFill(&sContext, &rect);
// рисуем рабочий диапазон
rect.sXMin = rect.sXMax;    //рабочий диапазон начинается после левой границы
rect.sXMax =KoordR0 - LengthRmin;
GrContextForegroundSet(&sContext, ClrLightGrey);
GrRectFill(&sContext, &rect);
// рисуемправуюграницу
 rect.sXMin = rect.sXMax;
rect.sXMax = rect.sXMin + LengthRmin;
GrContextForegroundSet(&sContext, ClrRed);
GrRectFill(&sContext, &rect);
//---------Рисуемточку-----------------------------------------------------------------------------
 rect.sXMin = pWidget->sPosition.sXMin;
 rect.sXMax =rect.sXMax;
  GrContextClipRegionSet(&sContext, &rect);
x = (bar_width - 1)*(R/RMax);*/
if (LengthR+.5 < POINT_RADIUS)
        LengthR = POINT_RADIUS;

if (LengthR+.5 > bar_width - POINT_RADIUS - 1)
        LengthR = bar_width - POINT_RADIUS - 1;

GrContextForegroundSet(&sContext, ClrBlack);

GrCircleFill(&sContext, sContext.sClipRegion.sXMax - (unsigned long)(LengthR+.5),
```

```
                sContext.sClipRegion.sYMin + BAR_HEIGHT / 2, POINT_RADIUS);
}

long Zone_Vilet_msg_proc(tWidget *pWidget, unsigned long ulMsg,
                         unsigned long ulParam1, unsigned long ulParam2)
{
    unsigned char i;
    switch(ulMsg) {
    case WIDGET_MSG_VALUE:
            for (i = 0; i < ARRAY_SIZE(Zone_Vilet_param_places); i++) {
                    if (Zone_Vilet_param_places[i].id == ulParam1) {
                            if (Zone_Vilet_values[i] != ulParam2) {
                                    Zone_Vilet_values[i] = ulParam2;
                                    if (WidgetIsInTree(WIDGET_ROOT, pWidget))
                                            Zone_Vilet_param_draw(pWidget, i);
                            }
                            break;
                    }
            }
            return 1;
    default:
            return CanvasMsgProc(pWidget, ulMsg, ulParam1, ulParam2);
    }
}
void Zone_Vilet_draw(tWidget *pWidget, tContext *pContext)
{
    unsigned char  i;
    pWidget->pfnMsgProc = Zone_Vilet_msg_proc;
```

```
    for (i = 0; i < ARRAY_SIZE(Zone_Vilet_param_places); i++)
            Zone_Vilet_param_draw(pWidget, i);
}
```

ПРИЛОЖЕНИЕВ

Код виджета отображения положения стрелы в зоне работы

```
#include "global_includes.h"
#include "grlib/grlib.h"
#include "grlib/widget.h"
#include "driverlib/debug.h"
#include "gui/gui.h"
#include "limiter/limiter.h"
#include "gui/lib/fatline.h"

#include "utils/sine.h"
#include "gui/lib/v_fp1.h"
#include "Arrow_draw.h"
#include "Arrowwidget.h"
#include "Highwidget.h"
#include "fstore/fstore_pbl_config.h"
//*****************************************************************************
//
// ArrowPaintBack
//
// Рисование неизменной части указателя
// &Zone4 - картинка зоны работы
//*****************************************************************************

static void
ArrowPaintBack(tWidget *pWidget)
{
```

```
        tRectangle *rect = &pWidget->sPosition;
    tArrowWidget *pArrowWidget = (tArrowWidget*)pWidget;
        tRectangle clip;
        tContext sContext;
        clip = *rect;
        GrContextInit(&sContext, pWidget->pDisplay);
        GrContextClipRegionSet(&sContext, &clip);
//      Перерисовываем изображение и удаляем предыдущую стрелку
// Draw image
        GrContextForegroundSet(&sContext,
                               pArrowWidget->ulBottomForegroundColor);
        GrContextBackgroundSet(&sContext,
                               pArrowWidget->ulBottomBackgroundColor);
    GrImageDraw(&sContext, &Zone4[0],
            rect->sXMin + (rect->sXMax - rect->sXMin + 1)*19/20 - GrImageWidthGet(&Zone4[0]) ,
                // Задаем от верхнего края странички, ниже которого будет располагаться картинка
            SHIFT_IMAGE_TOP);
}



//*****************************************************************************
//
// ArrowPaint
//
// Рисование изменяемой части указателя
//
//*****************************************************************************
extern const unsigned char AL_30_Grey[];
```

```
static void
ArrowPaint(tWidget *pWidget)
{
        tContext sContext;
        char string[64];
        tRectangle *rect = &pWidget->sPosition;
        tArrowWidget *pArrowWidget = (tArrowWidget*)pWidget;
        float angle, value2;
        unsigned short usX0, usY0, usXMax;
        long lRadius;
    long Lmax;     //Мфксимальная длина стрелы в отсчетах

        GrContextInit(&sContext, pWidget->pDisplay);
        GrContextClipRegionSet(&sContext, rect);

        //  value
        GrContextForegroundSet(&sContext,
                               pArrowWidget->ulBottomForegroundColor);
        GrContextBackgroundSet(&sContext,
                               pArrowWidget->ulBottomBackgroundColor);

        GrContextFontSet(&sContext, pArrowWidget->pFont);
                if ((pArrowWidget->ulValue) < 0x7FFF) {
                angle = pArrowWidget->ulValue;
                float fvalue = (unsigned long)angle;
                enum param_type etype = PARAM_FLOAT_0;
                if (pArrowWidget->ucFractLen == 1)
                        etype = PARAM_FLOAT_1;
```

```
                else if (pArrowWidget->ucFractLen == 2)
                        etype = PARAM_FLOAT_2;
                param2str(string, sizeof(string), &fvalue, etype, dig_sep, 0);
                strcat(string, " ");
                if (angle < pArrowWidget->ulMinValue)
                        angle = pArrowWidget->ulMinValue;
                if (angle > pArrowWidget->ulMaxValue)
                        angle = pArrowWidget->ulMaxValue;
        } else {
                angle = pArrowWidget->ulMinValue;
                usnprintf(string, sizeof(string), "---");
        }
        if ((pArrowWidget->lenValue) < 0x7FFF) {
                value2 = pArrowWidget->lenValue;
                float fvalue = (long)value2;
                enum param_type etype = PARAM_FLOAT_0;
                if (pArrowWidget->ucFractLen == 1)
                        etype = PARAM_FLOAT_1;
                else if (pArrowWidget->ucFractLen == 2)
                        etype = PARAM_FLOAT_2;
                param2str(string, sizeof(string), &fvalue, etype, dig_sep, 0);
                strcat(string, " ");
                if (value2 < (lim_config.length_y_array[0]-pbl_config.l_axis))    //length_y_array[0] -
Lmin
                        value2 = (lim_config.length_y_array[0]-pbl_config.l_axis);
                if (value2 > (lim_config.length_y_array[1]-pbl_config.l_axis))     //length_y_array[1] -
Lmax
                        value2 = (lim_config.length_y_array[1]-pbl_config.l_axis);
        } else {
```

```
                value2 = lim_config.length_y_array[0];
                usnprintf(string, sizeof(string), "---");
        }
    tRectangle rect_str;    //для передачи максимальных и минимальных значений в функцию рисования стрелы
        memcpy(&rect_str, rect, sizeof(tRectangle));
        tRectangle rect_src;
        tRectangle *prect_ctx = &sContext.sClipRegion;
        memcpy(&rect_src, prect_ctx, sizeof(tRectangle));
        // крайняялеваяточка - серединавиджета
        prect_ctx->sXMin = rect->sXMin + (rect->sXMax - rect->sXMin + 1)/2;
        // крайняя правая точка - учитываем скругление, чтобы его не нарушить
        prect_ctx->sXMax = rect->sXMax;
        // границы по вертикали равны границам нижней области
        prect_ctx->sYMin = rect->sYMax  + 3;
        prect_ctx->sYMax = rect->sYMax - 3;
        StringDrawAligned(&sContext, string, ALIGN_CENTER);
        memcpy(prect_ctx, &rect_src, sizeof(tRectangle));

        // для устранения мерцания ожидаем, когда контроллер дисплея дойдет до нижней строки контекста
        gui_lcd_line_wait(&sContext, rect->sYMax);
// Удаляемпредыдущуюстрелку
        GrContextForegroundSet(&sContext,
                               pArrowWidget->ulTopBackgroundColor);
        GrContextBackgroundSet(&sContext,
                               pArrowWidget->ulTopBackgroundColor);
        GrRectFill(&sContext, &pArrowWidget->sPointerArea);
        GrContextForegroundSet(&sContext,
```

```
                                pArrowWidget->ulBottomForegroundColor);
//----------------Рисуемизображение-------------------------------------------------------
// Перерисовываем изображение и удаляем предыдущую стрелку
// Draw image
        GrContextForegroundSet(&sContext,
                                pArrowWidget->ulBottomForegroundColor);
        GrContextBackgroundSet(&sContext,
                                pArrowWidget->ulBottomBackgroundColor);
    GrImageDraw(&sContext, &AL_30_Grey[0],
            rect->sXMin + (rect->sXMax - rect->sXMin + 1)*19/20 - GrImageWidthGet(&AL_30_Grey[0]) ,
                // Задаем от верхнего края странички, ниже которого будет располагаться картинка
SHIFT_IMAGE_TOP+GrImageHeightGet(&Zone4[0])-GrImageHeightGet(&AL_30_Grey[0]));
/*-----------------------------------------------------------------------------------------------
    Рисование границ зоны.
    (x1, y1) и (x2,y2)  - координаты начала и конца линий
-----------------------------------------------------------------------------------------------*/
    long x1, y1, x2, y2;
    GrContextForegroundSet(&sContext,
                                ClrRed);
        GrContextBackgroundSet(&sContext,
                                ClrWhite);

    x1=rect->sXMin + (rect->sXMax - rect->sXMin + 1)*19/20 - SHIFT_ARROW_LEFT;
    y1=GrImageHeightGet(&Zone4[0]) + SHIFT_IMAGE_TOP - SHIFT_ARRROW_BOTTOM;
    x2=(long)(x1-(GrImageHeightGet(&Zone4[0])-
SHIFT_ARRROW_BOTTOM)*cos(75*3.14/180)/sin(75*3.14/180));
    y2=rect->sYMin;
   GrFatLineDraw(&sContext, x1, y1, x2, y2, 4);
    x2=rect->sXMin + (rect->sXMax - rect->sXMin + 1)*19/20 - GrImageWidthGet(&Zone4[0])+24;
```

```
y2=(long)(y1+(x1-x2)*sin(7*3.14/180)/cos(7*3.14/180));
GrFatLineDraw(&sContext, x1, y1, x2, y2, 4);
      x1=x2;
y1=y2;
x2=x2;
y2=140;
GrFatLineDraw(&sContext, x1, y1, x2, y2, 4);
      x1=x1+50;
y1=y1-7;
x2=x1;
y2=y2-45;
GrFatLineDraw(&sContext, x1, y1, x2, y2, 4);

/*----------------------------------------------------------------------------------------------------------------------------
----------
    Вычисление данных положения стрелы
----------------------------------------------------------------------------------------------------------------------------
----------*/
        // центр стрелки
        usX0 =   rect->sXMin + (rect->sXMax - rect->sXMin + 1)*19/20 - SHIFT_ARROW_LEFT; //-
GrImageWidthGet(&Zone4[0])/2;
    usY0 = GrImageHeightGet(&Zone4[0]) + SHIFT_IMAGE_TOP - SHIFT_ARRROW_BOTTOM;

    usXMax=usX0-GrImageWidthGet(&Zone4[0])+SHIFT_ARROW_LEFT;

    Lmax=(long)((usY0 - rect->sYMin)/cos(15*3.14/180));
    // длинастрелки
    lRadius = (long)(value2*Lmax/(lim_config.length_y_array[1]-pbl_config.l_axis))-
3;//lim_config.length_y_array[1] - lmax; -3 - корректировкадляизбежаниязатиранияlинии
```

```
    // выводим единицу измерения если она есть
    /*if (pArrowWidget->ulUnit) {
            GrContextFontSet(&sContext, (tFont *)g_pucFont20pt);

            GrContextForegroundSet(&sContext,
                                   pArrowWidget->ulBottomForegroundColor);
            GrContextBackgroundSet(&sContext,
                                   pArrowWidget->ulTopBackgroundColor);

            prect_ctx->sXMin = rect->sXMin - (rect->sXMax - rect->sXMin) - 40;
            prect_ctx->sXMax = rect->sXMax - (rect->sXMax - rect->sXMin)/2 + 80;
            prect_ctx->sYMin = rect->sYMin-199;
            prect_ctx->sYMax = rect->sYMax-200;

            GrStringGet(pArrowWidget->ulUnit, string, sizeof(string));
            StringDrawAligned(&sContext, string, ALIGN_CENTER);
            memcpy(prect_ctx, &rect_src, sizeof(tRectangle));
    }*/
    // определяемуголстрелки
    /*long ulMinAngle, ulMaxAngle;

ulMinAngle = (unsigned long)(0xFFFFFFFF*(pArrowWidget->ulMinValue));
ulMaxAngle = (unsigned long)(0xFFFFFFFF*(pArrowWidget->ulRangeAngle));

if (angle<ulMinAngle) angle=ulMinAngle;
if (angle>ulMaxAngle) angle=ulMaxAngle;*/

    // рисуемстрелку
```

```
        GrContextForegroundSet(&sContext, pArrowWidget->ulTopForegroundColor);
        GrContextBackgroundSet(&sContext, pArrowWidget->ulTopBackgroundColor);
        Arrow_draw(&sContext, &pArrowWidget->sPointerArea, usXMax,
                usX0, usY0, angle, lRadius);
}
long
ArrowProc(tWidget *pWidget, unsigned long ulMsg,
          unsigned long ulParam1, unsigned long ulParam2)
{
        signed long value;
        unsigned char need_redraw = 0;
        unsigned char need_redraw_back = 0;

        ASSERT(pWidget);
        switch(ulMsg) {
        case WIDGET_MSG_PAINT: {
                ArrowPaintBack(pWidget);
                ArrowPaint(pWidget);
                return(1);
    }
        case WIDGET_MSG_VALUE: {
                value = *(signed long *)&ulParam2;
         float value2=*(float *)&ulParam2;
                if (ulParam1 == ARROW_VALUE) {
                        if (((tArrowWidget*)pWidget)->ulValue != value2) {
                                ((tArrowWidget*)pWidget)->ulValue = (signed long)value2;
                                need_redraw = 1;
                        }
```

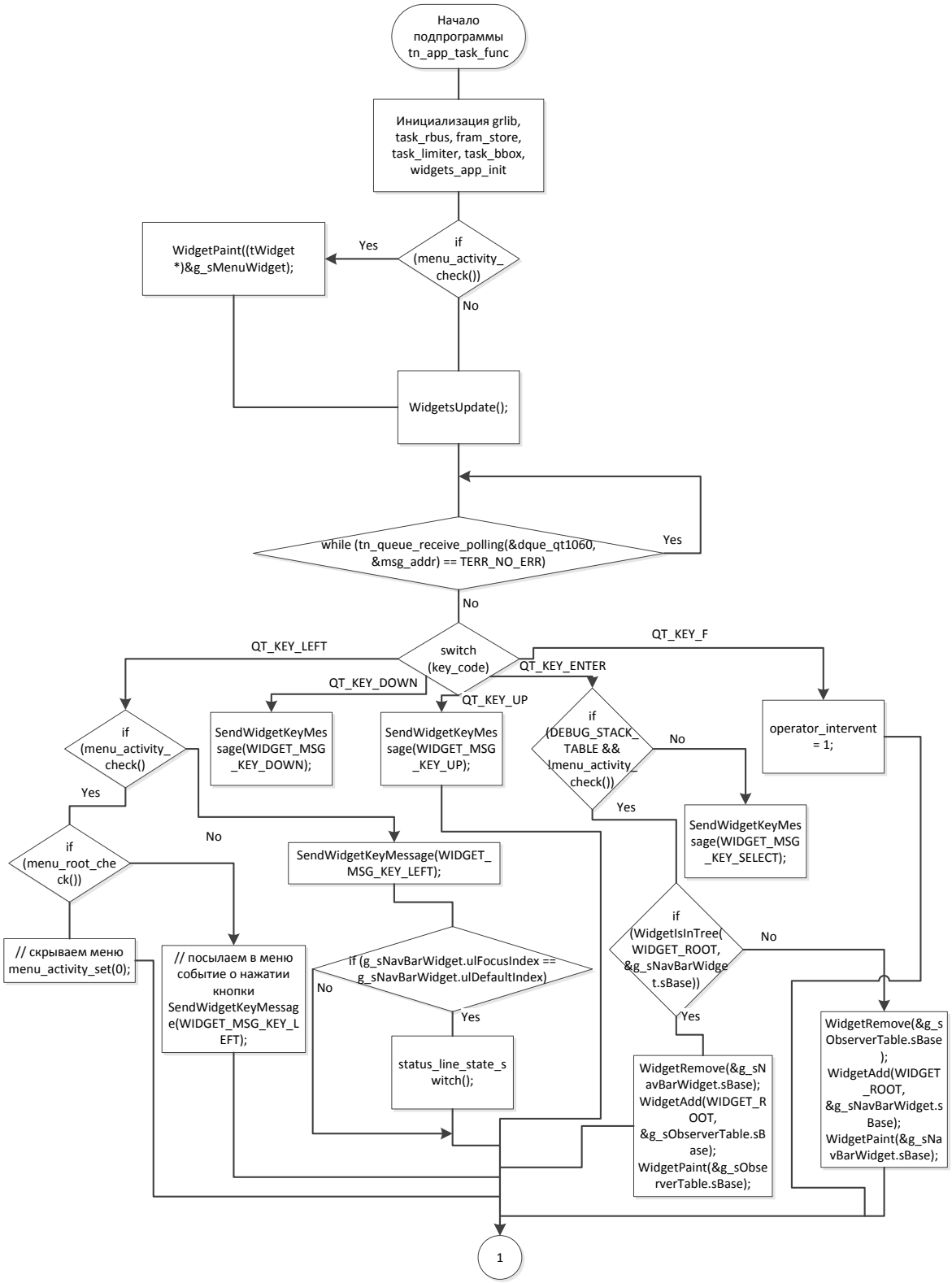
```
        } else    if (ulParam1 == LEN_STR_VALUE) {
                  if (((tArrowWidget*)pWidget)->lenValue != value2) {
                        ((tArrowWidget*)pWidget)->lenValue = value2;
                        need_redraw = 1;
                  }
      } else   if (ulParam1 == ARROW_MINVALUE) {
                  if (((tArrowWidget*)pWidget)->ulMinValue != value) {
                        ((tArrowWidget*)pWidget)->ulMinValue = value;
                        need_redraw = 1;
                        need_redraw_back = 1;
                  }
        } else    if (ulParam1 == ARROW_MAXVALUE) {
                  if (((tArrowWidget*)pWidget)->ulMaxValue != value) {
                        ((tArrowWidget*)pWidget)->ulMaxValue = value;
                        need_redraw = 1;
                        need_redraw_back = 1;
                  }

        } else    if (ulParam1 == ARROW_UNIT) {
                  if (((tArrowWidget*)pWidget)->ulUnit != value) {
                        ((tArrowWidget*)pWidget)->ulUnit = value;
                        // при смене единиц как правило изменится и значение,
                        // поэтому обновление виджета произойдёт автоматически
//                      need_redraw = 1;
//                      need_redraw_back = 1;
                  }
        }
```

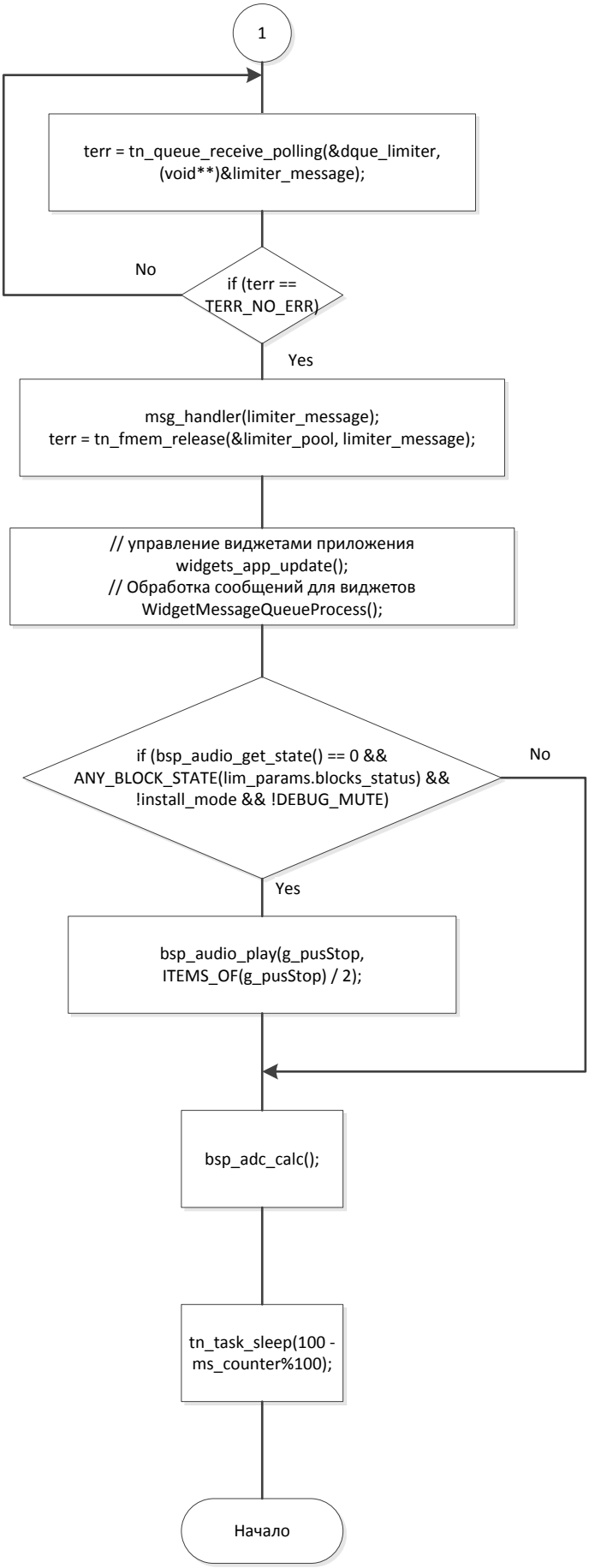
```
                if (need_redraw && WidgetIsInTree(WIDGET_ROOT, pWidget)) {
                        if (need_redraw_back) {
                                ArrowPaintBack(pWidget);
                        }
                        ArrowPaint(pWidget);
                        // check child
                        //
                        // See if this widget has a child.
                        //
                        if(pWidget->pChild)
                        {
                                //
                                // Go to this widget's child first.
                                //
                                // msg to text canvas
                                WidgetMessageQueueAdd(pWidget->pChild, WIDGET_MSG_UPDATE, 0,
0, 0, 0);
                        }
                }
                return(1);
        }
        default:
                return(WidgetDefaultMsgProc(pWidget, ulMsg,
                                        ulParam1, ulParam2));
        }

}
```

ПРИЛОЖЕНИЕ Г

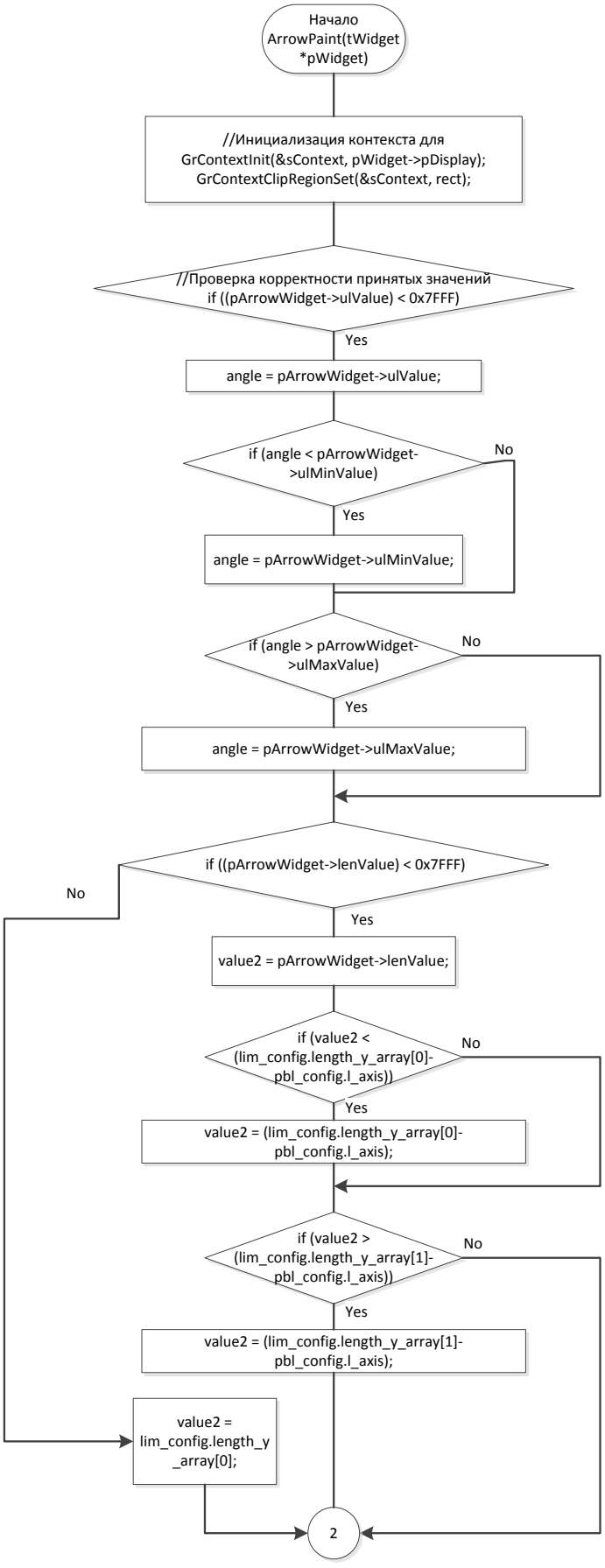
Разработанные алгоритмы

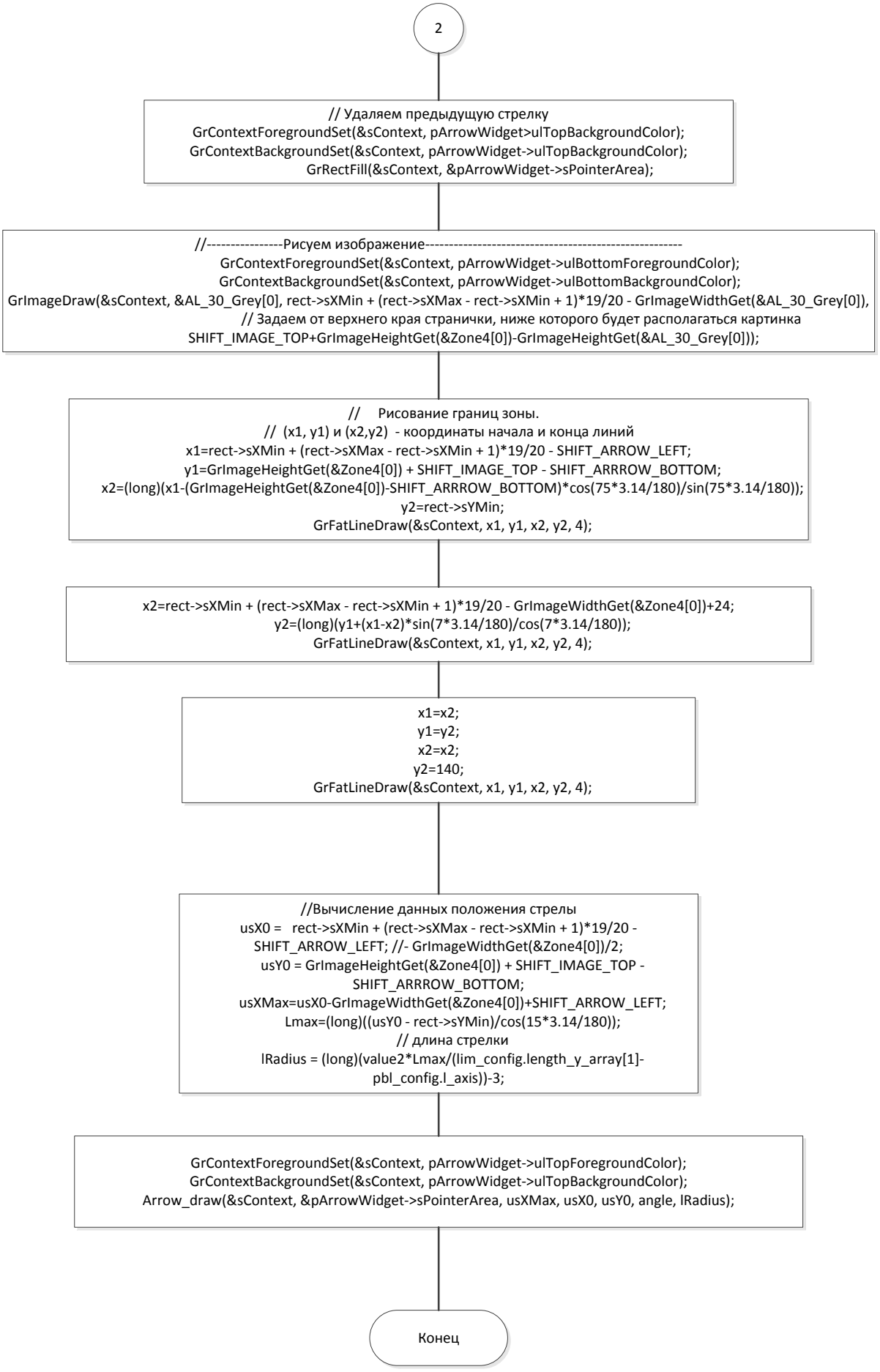
Г1 Алгоритм основного цикла




1
terr = tn_queue_receive_polling(&dque_limiter, (void**)&limiter_message);
No
if (terr == TERR_NO_ERR)
Yes
msg_handler(limiter_message);
terr = tn_fmem_release(&limiter_pool, limiter_message);
// управление виджетами приложения
widgets_app_update();
// Обработка сообщений для виджетов
WidgetMessageQueueProcess();
if (bsp_audio_get_state() == 0 && ANY_BLOCK_STATE(lim_params.blocks_status) && !install_mode && !DEBUG_MUTE)
No
Yes
bsp_audio_play(g_pusStop, ITEMS_OF(g_pusStop) / 2);
bsp_adc_calc();
tn_task_sleep(100 - ms_counter%100);
Начало

Г2 Алгоритм страницы с зоной работы




2
// Удаляем предыдущую стрелку
GrContextForegroundSet(&sContext, pArrowWidget>ulTopBackgroundColor);
GrContextBackgroundSet(&sContext, pArrowWidget->ulTopBackgroundColor);
GrRectFill(&sContext, &pArrowWidget->sPointerArea);
//----------Рисуем изображение----------
GrContextForegroundSet(&sContext, pArrowWidget->ulBottomForegroundColor);
GrContextBackgroundSet(&sContext, pArrowWidget->ulBottomBackgroundColor);
GrImageDraw(&sContext, &AL_30_Grey[0], rect->sXMin + (rect->sXMax - rect->sXMin + 1)*19/20 - GrImageWidthGet(&AL_30_Grey[0]),
// Задаем от верхнего края странички, ниже которого будет располагаться картинка
SHIFT_IMAGE_TOP+GrImageHeightGet(&Zone4[0])-GrImageHeightGet(&AL_30_Grey[0]));
// Рисование границ зоны.
// (x1, y1) и (x2,y2) - координаты начала и конца линий
x1=rect->sXMin + (rect->sXMax - rect->sXMin + 1)*19/20 - SHIFT_ARROW_LEFT;
y1=GrImageHeightGet(&Zone4[0]) + SHIFT_IMAGE_TOP - SHIFT_ARRROW_BOTTOM;
x2=(long)(x1-(GrImageHeightGet(&Zone4[0])-SHIFT_ARRROW_BOTTOM)*cos(75*3.14/180)/sin(75*3.14/180));
y2=rect->sYMin;
GrFatLineDraw(&sContext, x1, y1, x2, y2, 4);
x2=rect->sXMin + (rect->sXMax - rect->sXMin + 1)*19/20 - GrImageWidthGet(&Zone4[0])+24;
y2=(long)(y1+(x1-x2)*sin(7*3.14/180)/cos(7*3.14/180));
GrFatLineDraw(&sContext, x1, y1, x2, y2, 4);
x1=x2;
y1=y2;
x2=x2;
y2=140;
GrFatLineDraw(&sContext, x1, y1, x2, y2, 4);
//Вычисление данных положения стрелы
usX0 = rect->sXMin + (rect->sXMax - rect->sXMin + 1)*19/20 -
SHIFT_ARROW_LEFT; //- GrImageWidthGet(&Zone4[0])/2;
usY0 = GrImageHeightGet(&Zone4[0]) + SHIFT_IMAGE_TOP -
SHIFT_ARRROW_BOTTOM;
usXMax=usX0-GrImageWidthGet(&Zone4[0])+SHIFT_ARROW_LEFT;
Lmax=(long)((usY0 - rect->sYMin)/cos(15*3.14/180));
// длина стрелки
lRadius = (long)(value2*Lmax/(lim_config.length_y_array[1]-
pbl_config.l_axis))-3;
GrContextForegroundSet(&sContext, pArrowWidget->ulTopForegroundColor);
GrContextBackgroundSet(&sContext, pArrowWidget->ulTopBackgroundColor);
Arrow_draw(&sContext, &pArrowWidget->sPointerArea, usXMax, usX0, usY0, angle, lRadius);
Конец

ПРИЛОЖЕНИЕ Д

Схема тестирования универсального блока индикации