МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования «Южно-Уральский государственный университет»
(национальный исследовательский университет)
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2024 г.

Разработка драйвера для датчика давления

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2024.406 ПЗ ВКР

Руководитель работы,
к.п.н., доцент каф. ЭВМ
_____ Ю.Г. Плаксина
«___»_____ 2024 г.

Автор работы,
студент группы КЭ-406
_____ М.М. Горшенин
«___»_____ 2024 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____ 2024 г.

Челябинск-2024

Аннотация

М.М. Горшенин. Разработка драйвера для датчика давления – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2024, 155 с., 56 ил., библиогр. список – 26 наим.

В рамках выпускной квалификационной работы проводится обзор принципов измерения давления, способов передачи информации, а также методов создания описания работы датчиков. Помимо этого, рассмотрена актуальность используемых технологий для обоснования разработки драйвера.

Работа включает в себя формулирование функциональных и нефункциональных требований и разработку на их основе архитектуры драйвера, используя нотацию IDEF0 (Integrated DEFinition).

Итогом проделанной работы является драйвер для датчика давления «Метран-150», разработанный на языке EDDL (Electronic Device Description Language, язык описания электронных устройств). Программный код прошел все необходимые тесты, для проведения которых использовалось программное обеспечение AMS Device Configurator – конфигуратор для интеграции устройств.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

На сегодняшний день предприятия, которые стремятся к модернизации, прибегают к промышленной автоматизации. Важность производственного автоматизирования понимают все участники рынка: и производители, и государство, которое способствует внедрению новых технологий [1]. Промышленная автоматизация представляет из себя систему управления, состоящую из различных устройств и информационных технологий для контроля производственных процессов. Целью автоматизирования является повышение производительности, уровня безопасности и качества продукции и снижение операционных затрат.

Программируемая система автоматизации является одним из подходов в промышленной оптимизации. В ней предполагается, что последовательность операций и конфигурация оборудования могут быть изменены с помощью электронных средств управления. Такая система требует значительного времени и ресурсов для проектирования и создания программного обеспечения и чаще всего используется в серийном производстве.

Как правило системы автоматизации имеют в своем составе измерительные датчики. Это могут быть как устройства, излучающие аналоговый сигнал, так и с дискретным сигналом (включение/выключение питания). Назначение измерительных датчиков – преобразовывать различные физические величины в электрические.

Среди измерительных средств существуют датчики давления полевого применения, которые могут применяться в самых различных областях промышленности. Это приборы, измеряющие давление, которое определяется как величина силы, действующей на определенную область. Датчики могут использоваться во множестве различных отраслях: нефтегазовая, химическая, горнодобывающая и пр. Для управления измерительными устройствами необходим драйвер.

Цель работы – разработать драйвер для измерительного датчика давления «Метран-150».

Актуальность темы – использование разработанного драйвера позволит с помощью коммуникатора производить необходимые манипуляции над датчиком давления: настройка, калибровка, диагностика и прочий функционал, связанный с его обслуживанием и эксплуатацией.

Задачи, поставленные для достижения цели:

1. Аналитический обзор научно-технической, нормативной и методической литературы по тематике работы.
2. Формулирование требований, разработка архитектуры драйвера.
3. Разработка и отладка программного кода.
4. Тестирование и доработка программного кода.

1. АНАЛИТИЧЕСКИЙ ОБЗОР НАУЧНО-ТЕХНИЧЕСКОЙ, НОРМАТИВНОЙ И МЕТОДИЧЕСКОЙ ЛИТЕРАТУРЫ ПО ТЕМАТИКЕ РАБОТЫ

Технологии, используемые для передачи информации об измерениях датчиков (интерфейсы и промышленные сети), напрямую влияют на создание драйвера (описания) устройства.

Драйвер будет разрабатываться для датчика, основанного на тензометрическом и емкостном методах измерения давления. Для создания драйвера устройства используется язык описания электронных устройств (EDDL).

1.1. Принцип работы датчиков давления

Датчик давления состоит из первичного преобразователя давления, в составе которого чувствительный элемент и приемник давления, схемы вторичной обработки сигнала, различных по конструкции корпусных деталей и устройства вывода (рисунок 1.1). Основным отличием одних приборов от других является точность регистрации давления, которая зависит от принципа преобразования данных в электрический сигнал: тензометрический, пьезорезистивный, емкостной, индуктивный, резонансный, ионизационный.

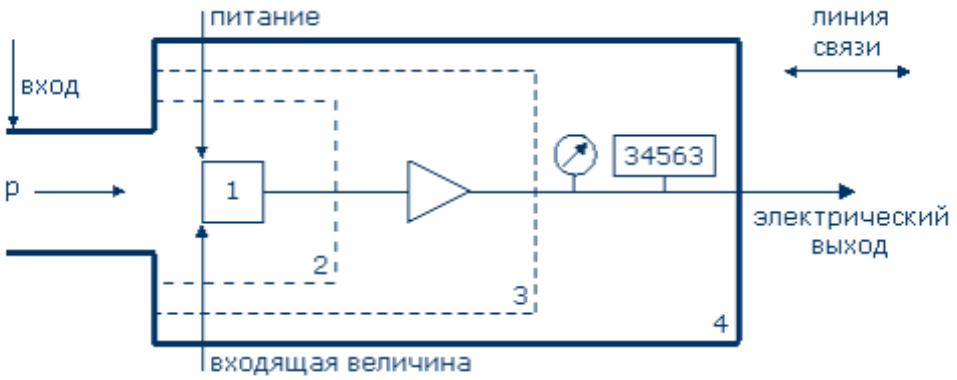

Рисунок 1.1 – Преобразователь давления в электрический сигнал

Рассмотрим более подробно методы преобразования давления:

1.1.1. Тензометрический метод

В настоящее время большая часть датчиков давления производится на основе чувствительных элементов [2] (рисунок 1.2), принцип работы которых заключается в измерении деформации тензорезисторов, сформированных в эпитаксиальной пленке кремния на подложке из сапфира (КНС), припаянной твердым припоем к титановой мембране. Иногда вместо кремниевых тензорезисторов применяются металлические: медные, никелевые, железные и другие.

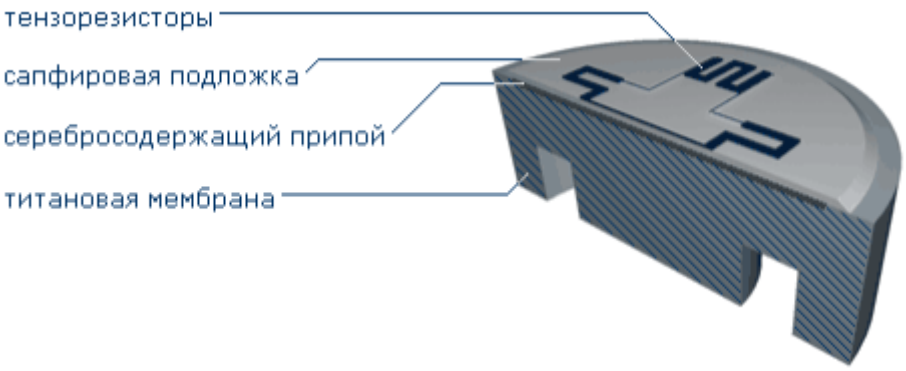

Рисунок 1.2 – Тензорезистивный чувствительный элемент

Тензопреобразователи работают на основе явления тензоэффекта в материалах. Чувствительным элементом служит мембрана с тензорезисторами, соединенными в мостовую схему. Под действием давления измеряемой среды мембрана прогибается, тензорезисторы меняют свое сопротивление, что приводит к разбалансу моста Уитстона. Разбаланс линейно зависит от степени деформации резисторов и, следовательно, от приложенного давления.

Следует отметить принципиальное ограничение КНС преобразователя – неустранимую временную нестабильность градуировочной характеристики и существенные гистерезисные эффекты от давления и температуры. Это обусловлено неоднородностью конструкции и жесткой связью мембраны с

конструктивными элементами датчика. Поэтому, выбирая преобразователь на основе КНС, необходимо обратить внимание на величину основной погрешности с учетом гистерезиса и величину дополнительной погрешности.

К преимуществам можно отнести хорошую защищенность чувствительного элемента от воздействия любой агрессивной среды, налаженное серийное производство, низкую стоимость.

1.1.2. Пьезорезистивный метод

Многие производители датчиков проявляют живой интерес к использованию интегральных чувствительных элементов на основе монокристаллического кремния [3]. Это обусловлено тем, что кремниевые преобразователи имеют на порядок большую временную и температурную стабильности по сравнению с приборами на основе КНС (кремний на сапфире) структур.

Кремниевый интегральный преобразователь давления (ИПД, рисунок 1.3) представляет собой мембрану из монокристаллического кремния с диффузионными пьезорезисторами, подключенными в мост Уинстона. Чувствительным элементом служит кристалл ИПД, установленный на диэлектрическое основание с использованием легкоплавкого стекла или методом анодного сращивания [4, 5].

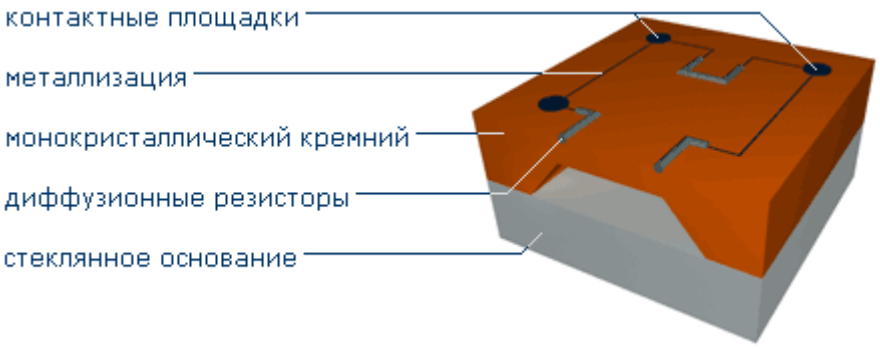

Рисунок 1.3 – Кремниевый интегральный преобразователь давления

Для измерения давления чистых неагрессивных сред применяются, так

называемые, Low cost – решения (рисунок 1.4), основанные на использовании чувствительных элементов либо без защиты, либо с защитой силиконовым гелем [5].

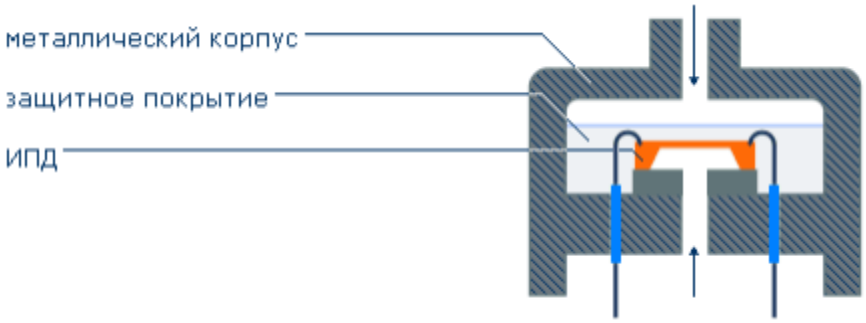

Рисунок 1.4 – Low Cost решение для пьезорезистивных чувствительных элементов

Для измерения агрессивных сред и большинства промышленных применений применяется преобразователь давления в герметичном металлостеклянном корпусе, с разделительной диафрагмой из нержавеющей стали, передающей давление измеряемой среды на ИПД посредством кремнийорганической жидкости (рисунок 1.5).

Рисунок 1.5 – Преобразователь давления, защищенный от измеряемой среды посредством коррозионностойкой мембраны

Пьезорезистивные датчики обладают основным преимуществом – более высокой стабильностью характеристик, по сравнению с КНС преобразователями. ИПД на основе монокристаллического кремния устойчивы к воздействию ударных и знакопеременных нагрузок. Если не происходит механического разрушения чувствительного элемента, то после снятия нагрузки он возвращается к первоначальному состоянию, что объясняется использованием идеально-упругого материала [5].

1.1.3. Емкостной метод

Емкостные преобразователи используют метод изменения емкости конденсатора при изменении расстояния между обкладками. Известны керамические или кремниевые емкостные первичные преобразователи давления и преобразователи, выполненные с использованием упругой металлической мембраны [5]. При изменении давления мембрана с электродом деформируется и происходит изменение емкости. В элементе из керамики или кремния, пространство между обкладками обычно заполнено маслом или другой органической жидкостью (рисунок 1.6) [6].

Рисунок 1.6 – Емкостной керамический преобразователь давления, выполненный методами микромеханики

При использовании металлической диафрагмы (рисунок 1.7) ячейка делится на две части, с одной стороны которой расположены электроды. Электроды с диафрагмой образуют две переменные емкости, включенные в плечи измерительного моста. Когда давление по обеим сторонам одинаково, мост сбалансирован. Изменение давления в одной из камер приводит к деформации мембраны, что изменяет емкости, разбалансируя мост. В настоящее время сенсоры изготавливаются с электродами, расположенными с одной стороны от диафрагмы. Газ будет контактировать только с камерой, выполненной из нержавеющей стали или инконеля. Это позволяет проводить измерения давления загрязненных, агрессивных, радиоактивных газов и смесей неизвестного состава. В абсолютной модели опорное давление составляет 10-7 – 10-8 мм рт. ст., которое поддерживается в течение длительного времени химическим геттером.

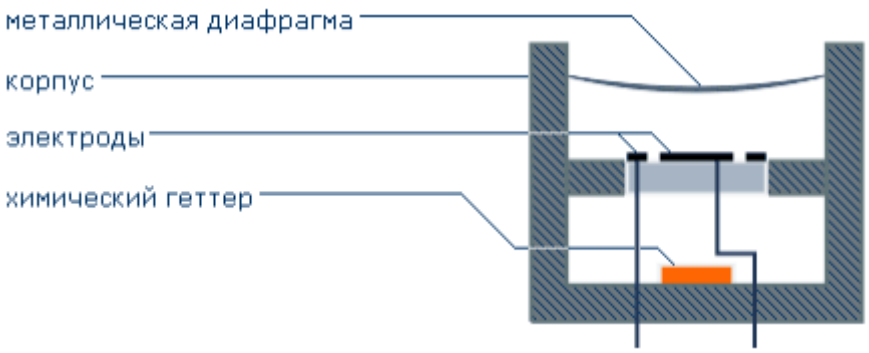

Рисунок 1.7 – Емкостной преобразователь давления с металлической диафрагмой в роль подвижной обкладки конденсатора

Достоинством чувствительного емкостного элемента является простота конструкции, высокая точность и временная стабильность, возможность измерять низкие давления и слабый вакуум. К недостатку можно отнести нелинейную зависимость емкости от приложенного давления.

1.1.4. Резонансный метод

Резонансный принцип используется в датчиках давления на основе вибрирующего цилиндра, струнных датчиках, кварцевых датчиках, резонансных датчиках на кремнии. В основе метода лежат волновые процессы: акустические или электромагнитные. Это и объясняет высокую стабильность датчиков и высокие выходные характеристики прибора [6, 7].

Частным примером может служить кварцевый резонатор (рисунок 1.8). При прогибе мембраны, происходит деформация кристалла кварца, подключенного в электрическую схему, и его поляризация. В результате изменения давления частота колебаний кристалла меняется. Подобрав параметры резонансного контура, изменяя емкость конденсатора или индуктивность катушки, можно добиться того, что сопротивление кварца падает до нуля – частоты колебаний электрического сигнала и кристалла совпадают – наступает резонанс.

Рисунок 1.8 – Упрощенный вид резонансного чувствительного элемента, выполненного на кварце

Резонансные датчики обладают высокой точностью и стабильностью характеристик, которые зависят от качества используемого материала. Однако у них есть и недостатки, такие как индивидуальная характеристика преобразования давления, значительное время отклика и невозможность проводить измерения в агрессивных средах без потери точности показаний прибора [8].

1.1.5. Индуктивный метод

Индукционный метод измерения давления основан на регистрации вихревых токов (токов Фуко). Чувствительный элемент состоит из двух катушек, изолированных между собой металлическим экраном (рисунок 1.9). Преобразователь измеряет смещение мембраны при отсутствии механического контакта [9]. В катушках генерируется электрический сигнал переменного тока таким образом, что заряд и разряд катушек происходит через одинаковые промежутки времени. При отклонении мембраны создается ток в фиксированной основной катушке, что приводит к изменению индуктивности системы. Смещение характеристик основной катушки дает возможность преобразовать давление в стандартизированный сигнал, по своим параметрам прямо пропорциональный приложенному давлению.

Рисунок 1.9 – Принципиальная схема индукционного преобразователя давления

Преимуществом такой системы, является возможность измерения низких избыточных и дифференциальных давлений, достаточно высокая точность и незначительная температурная зависимость. Однако датчик чувствителен к магнитным воздействиям, что объясняется наличием катушек, которые при прохождении переменного сигнала создают магнитное поле.

1.1.6. Ионизационный метод

В основе лежит принцип регистрации потока ионизированных частиц. Аналогом являются ламповые диоды (рисунок 1.10).

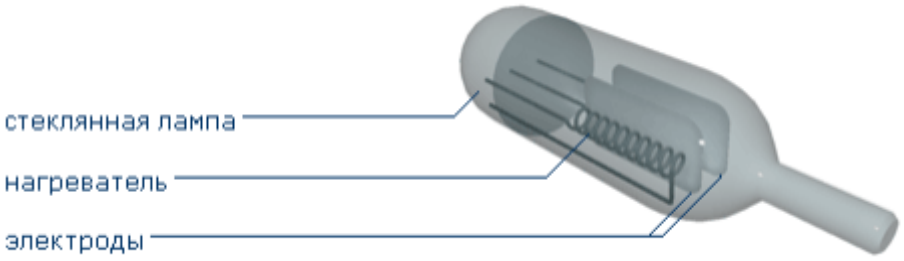

Рисунок 1.10 – Ионизационный датчик вакуума

Лампа оснащена двумя электродами: катодом и анодом, – а также нагревателем [6, 7]. В некоторых лампах последний отсутствует, что связано с использованием более совершенных материалов для электродов. Корпус лампы выполнен из высококачественного стекла.

Преимуществом таких ламп является возможность регистрировать низкое давление – вплоть до глубокого вакуума с высокой точностью. Однако следует

строго учитывать, что подобные приборы нельзя эксплуатировать, если давление в камере близко к атмосферному. Поэтому подобные преобразователи необходимо сочетать с другими датчиками давления, например, емкостными. Помимо прочего, ионизационные лампы должны оснащаться дополнительными приборами, поскольку зависимость сигнала от давления является логарифмической.

Различные сферы применений определяют свои требования к датчикам: для промышленности – надежность и стабильность характеристик, для лабораторных измерений и расходометрии – точность измерения давления и т.д. Еще одним важным параметром является цена датчиков, которые используют тот или иной принцип преобразования давления. Поэтому при выборе преобразователя необходимо определить наиболее выгодный вариант – соотношение цены к возможностям прибора. Очевидно там, где требуется только какой-либо определенный параметр датчика (например, точность или возможность измерять вакуум) соотношение цены к предъявляемым требованиям высокое. В основном это касается резонансных, индуктивных, емкостных и ионизационных датчиков.

В большинстве случаев требуется несколько параметров преобразователей: точность, стабильность выходных характеристик, надежность, долговечность, низкая цена. Таким требованиям, как видно из вышеприведенной таблицы, удовлетворяют пьезорезистивные датчики давления и КНС-преобразователи. Выбрав КНС-преобразователи, вы получите надежные датчики, работающие при высоких температурах (более 1500С), однако теряете в точности и стабильности выходных характеристик, по сравнению с преобразователями на монокристаллическом кремнии. Поскольку в основном требуется высокая стабильность выходных характеристик при невысоких температурах, то интегральные преобразователи давления являются в этом случае оптимальным решением, при невысокой цене.

1.2. Промышленные сети

Промышленной сетью называют комплекс оборудования и программного обеспечения, которые обеспечивают обмен информацией (коммуникацию) между несколькими устройствами. Промышленная сеть является основой для построения распределенных систем сбора данных и управления.

Далее рассмотрим существующие варианты промышленных сетей.

1.2.1. Полевые шины

Главной функцией полевой шины является обеспечение сетевого взаимодействия между контроллерами и удаленной периферией (например, узлами ввода/вывода). Помимо этого, к полевой шине могут подключаться различные контрольно-измерительные приборы и исполнительные (полевые) устройства, снабженные соответствующими сетевыми интерфейсами.

Принцип сетевого обмена данными основан на классическом принципе «ведущий-ведомый» (master-slave) или его небольших модификациях. Современные полевые шины удовлетворяют строгим техническим требованиям, благодаря чему становится возможной их эксплуатация в тяжелых промышленных условиях. К этим требованиям относятся:

1. Детерминированность – передача сообщения из одного узла сети в другой занимает строго фиксированный отрезок времени. Офисные сети, построенные по технологии Ethernet, — это отличный пример недетерминированной сети. Сам алгоритм доступа к разделяемой среде по методу CSMA/CD не определяет время, за которое кадр из одного узла сети будет передан другому, и, строго говоря, нет никаких гарантий, что кадр вообще дойдет до адресата. Для промышленных сетей это недопустимо. Время передачи сообщения должно быть ограничено и в общем случае, с учетом количества узлов, скорости передачи данных и длины сообщений, может быть заранее рассчитано.

2. Поддержка больших расстояний – существенное требование, так как расстояние между объектами управления может достигать нескольких километров. Применяемый протокол должен быть ориентирован на использование в сетях большой протяженности.

3. Защита от электромагнитных наводок. Длинные линии в особенности подвержены пагубному влиянию электромагнитных помех, излучаемых различными электрическими агрегатами. Сильные помехи в линии могут исказить передаваемые данные до неузнаваемости. Для защиты от таких помех применяют специальные экранированные кабели, а также оптоволокно, которое, в силу световой природы информационного сигнала, вообще нечувствительно к электромагнитным наводкам. Кроме этого, в промышленных сетях должны использоваться специальные методы цифрового кодирования данных, препятствующие их искажению в процессе передачи или, по крайней мере, позволяющие эффективно детектировать искаженные данные принимающим узлом.

4. Упрочненная механическая конструкция кабелей и соединителей. Кабели и соединители должны быть прочными, долговечными и приспособленными для использования в самых тяжелых условиях (в том числе в агрессивных атмосферах, в условиях повышенного уровня вибраций, влажности).

По виду физической среды передачи данных полевые шины делятся на два типа:

1. Построенные на базе оптоволоконного кабеля. Преимущества использования оптоволокна: возможность построения протяженных коммуникационных линий (до 10 км и более), большая полоса пропускания, нечувствительность к электромагнитным помехам, возможность прокладки во взрывоопасных зонах. Недостатки: относительно высокая стоимость кабеля, сложность физического подключения и соединения кабелей.

2. Построенные на базе медного кабеля. Как правило, это двухпроводной кабель типа «витая пара» со специальной изоляцией и экранированием.

Преимущества: доступная цена, легкость прокладки и выполнения физических соединений. Недостатки: подвержен влиянию электромагнитных наводок, ограниченная протяженность кабельных линий, меньшая по сравнению с оптоволокном полоса пропускания [10].

1.2.2. Промышленный интернет

Применению промышленного интернета долгое время мешал метод случайного доступа к сети, не гарантирующий доставку сообщения в короткое и заранее известное время. Однако эта проблема была решена применением коммутаторов, которые обеспечивают детерминированное поведение сетей [11].

Недостатком промышленного Ethernet является относительно высокая цена: модули ввода-вывода в среднем в 2 раза дороже аналогичных полевых устройств. К преимуществам можно отнести:

1. Высокая скорость передачи (до 10 Гбит/с) и соответствие требованиям жесткого реального времени при высоком быстродействии (например, при управлении движением).
2. Простота интеграции с офисными сетями.
3. Возможность организации многомастерных сетей.
4. Неограниченные возможности по организации сетей самых разнообразных топологий [12].

1.2.3. Беспроводные сети

Существует множество объектов автоматизации, применение беспроводных сетей является крайне желательным:

1. Устройства на подвижных частях (конвейеры, лифты и тп).
2. Объекты, в которых вносить капитальные изменения в конструкцию здания (датчики для систем обогрева и кондиционирования воздуха, системы

умного дома).

3. Объекты с агрессивными средами, вибрацией; объекты, находящиеся под высоким напряжением или в местах, неудобных для прокладки кабеля.

4. Отслеживание траектории движения транспорта, охрана границ государства, мониторинг напряженности автомобильного трафика в городах и условий на дорогах, мониторинг леса, моря, сельскохозяйственных культур, вредных выбросов.

5. Объекты во взрывоопасных зонах.

Беспроводные сети позволяют достичь следующих преимуществ по сравнению с проводными сетями:

1. Снижение стоимости установки датчиков.
2. Исключение необходимость профилактического обслуживания кабелей.
3. Исключение дорогостоящего места разветвлений кабеля.
4. Уменьшение трудозатратов и времени на монтаж и обслуживание системы.
5. Ускорение отладки системы и поиска неисправностей.
6. Обеспечение удобной модернизации системы.

С точки зрения требований к промышленным сетям беспроводные сети уступают проводным по следующим характеристикам:

1. Помехозащищенность: беспроводные сети подвержены влиянию электромагнитных помех значительно сильнее, чем проводные.

2. Надежность связи: связь может исчезнуть при несвоевременной замене батарей питания, изменении расположения узлов сети или появлении объектов, которые вызывают затухание, отражение, преломление и рассеяние радиоволн.

3. Ограниченная дальность связи без использования ретрансляторов (обычно не более 100 м внутри помещений).

4. Резкое падение пропускной способности сети при увеличении количества одновременно работающих станций и коэффициент использования канала.

5. Безопасность: возможность утечки информации, незащищенность от искусственно созданных помех, возможность незаметного управления технологическим процессом враждебными лицами [13].

1.3. Промышленные интерфейсы

Соединение промышленной сети с ее компонентами (устройствами, узлами сети) выполняется с помощью интерфейсов. Сетевым интерфейсом называют логическую и (или) физическую границу между устройством и средой передачи информации. Обычно этой границей является набор электронных компонентов и связанного с ними программного обеспечения.

1.3.1. Интерфейс RS-485

В основе построения интерфейса RS-485 лежит дифференциальный способ передачи сигнала, когда напряжение, соответствующее уровню логической единицы или нуля, отсчитывается не от «земли», а измеряется как разность потенциалов между двумя передающими линиями: Data+ и Data- (рисунок 1.11). При этом напряжение каждой линии относительно "земли" может быть произвольным, но не должно выходить за диапазон -7...+12 В.

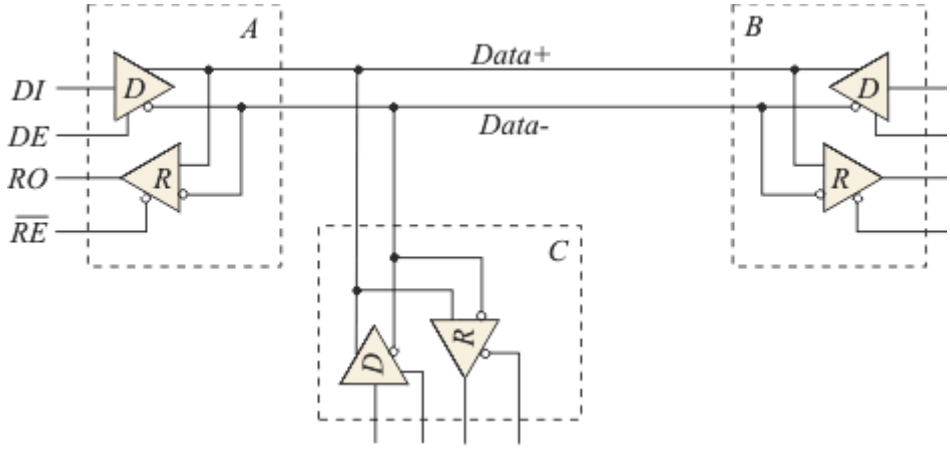

Рисунок 1.11 – Соединение трех устройств с интерфейсом RS-485 по двухпроводной схеме

Приемники сигнала являются дифференциальными, т.е. воспринимают

только разность между напряжениями на линии Data+ и Data-. При разности напряжений более 200 мВ, до +12 В считается, что на линии установлено значение логической единицы, при напряжении менее -200 мВ, до -7 В – логического нуля.

Благодаря симметрии линий относительно «земли» в них наводятся помехи, близкие по форме и величине. В приемнике с дифференциальным входом сигнал выделяется путем вычитания напряжений на линиях, поэтому после вычитания напряжение помехи оказывается равным нулю. В реальных условиях, когда существует небольшая асимметрия линий и нагрузок, помеха подавляется не полностью, но ослабляется существенно.

1.3.2. Интерфейс «токовая петля»

Принцип работы токовой петли заключается в передачи сигналов (информации) с помощью электрического тока (рисунок 1.12). Для измерительных датчиков, в том числе датчиков давления, применяется аналоговая токовая петля. Используемый диапазон – 4-20 мА, то есть наименьшее значение сигнала соответствует току 4 мА, а наибольшее — 20 мА [14]. Нулевое значение тока в цепи означает обрыв цепи и позволяет легко распознать критическую ситуацию. Также допустимо использовать сигналы вне диапазона для дополнительных индикаций [15].

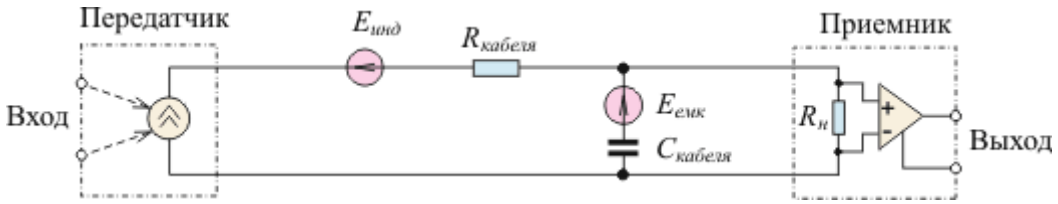

Рисунок 1.12 – Принцип действия токовой петли

Основным недостатком токовой петли является ее принципиально низкое быстродействие, которое ограничивается скоростью заряда емкости кабеля от источника тока. Например, при типовой погонной емкости кабеля 75 пФ/м и длине 1 км емкость кабеля составит 75 нФ. Для заряда такой емкости от

источника тока 20 мА до напряжения 5 В необходимо время 19 мкс, что соответствует скорости передачи около 9 кбит/с.

Однако токовая петля имеет преимущество, когда необходимо передавать сигналы от датчиков к контроллерам: она поддерживается большинством поставщиков устройств промышленной автоматизации, предоставляет возможность высокоточной (не более +-0,05% погрешности) передачи сигнала на значительное расстояние.

1.4. Промышленные протоколы

При создании нового поколения интеллектуальных приборов и датчиков потребовалось наряду с передачей аналоговой информации передавать и цифровые данные.

1.4.1. HART

В середине 80-х годов американская компания Rosemount разработала протокол Highway Addressable Remote Transducer (HART). В начале 90-х годов протокол был дополнен и стал открытым коммуникационным стандартом [16]. Вначале он был нормирован только для применения в режиме соединения «точка-точка», затем появилась возможность применять протокол в режиме многоточечного соединения («multidrop»). Основные технические параметры, определяемые стандартом на HART-протокол, представлены в таблице 1 [17].

Таблица 1 – Описание технических параметров HART-протокола

Параметр	Значение
Максимальное кол-во устройств	Одно подчиненное устройство и два ведущих устройства (стандартный режим); 15 подчиненных устройств, 2 ведущих устройства (многоточечный режим с удаленным питанием)
Максимальная протяженность линии связи	3 км (стандарт); 100 м (многоточечный режим)
Тип линии	Экранированная витая пара
Интерфейс	4-20 мА, токовая петля (аналоговый)
Скорость передачи	1,2 кбит/с
Метод обращения	Polling (механизм опроса с уникальной адресацией каждого устройства)
Максимальная длина пакета данных	0-25 байт
Время цикла обновления данных	Около 500 мс (в пакетном режиме — 330 мс)
Надежность передачи данных	1 ошибка на 10^5 бит, контроль по четности каждого байта, байт контрольной суммы для каждого пакета
Возможность использования во взрывоопасной зоне	Да

HART-протокол используется в двух режимах подключения. В большинстве случаев применяется соединение «точка-точка» (рисунок 1.13), то есть непосредственное соединение прибора низовой автоматики (преобразователя информации, датчика, исполнительного устройства и т.п.) и не более чем двух ведущих устройств. В качестве первичного ведущего устройства, как правило, используется устройство связи с объектом (УСО) или программируемый логический контроллер, а в качестве вторичного — портативный HART-терминал или отладочный персональный компьютер с соответствующим модемом. При этом аналоговый токовый сигнал передается от ведомого прибора к соответствующему ведущему устройству. Цифровые сигналы могут приниматься или передаваться как от ведущего, так и от ведомого устройства. Так как цифровой сигнал наложен на аналоговый, процесс передачи аналогового сигнала происходит без прерывания [18]. В

многоточечном режиме (рисунок 1.14) до 15 ведомых устройств (slave) могут соединяться параллельно двухпроводной линией с теми же двумя ведущими устройствами (master). При этом по линии осуществляется только цифровая связь. Сигнал постоянного тока 4 мА обеспечивает вспомогательное питание ведомых приборов по сигнальным линиям. Типовые HART-компоненты и схема их подключения показаны на рисунке 1.15.

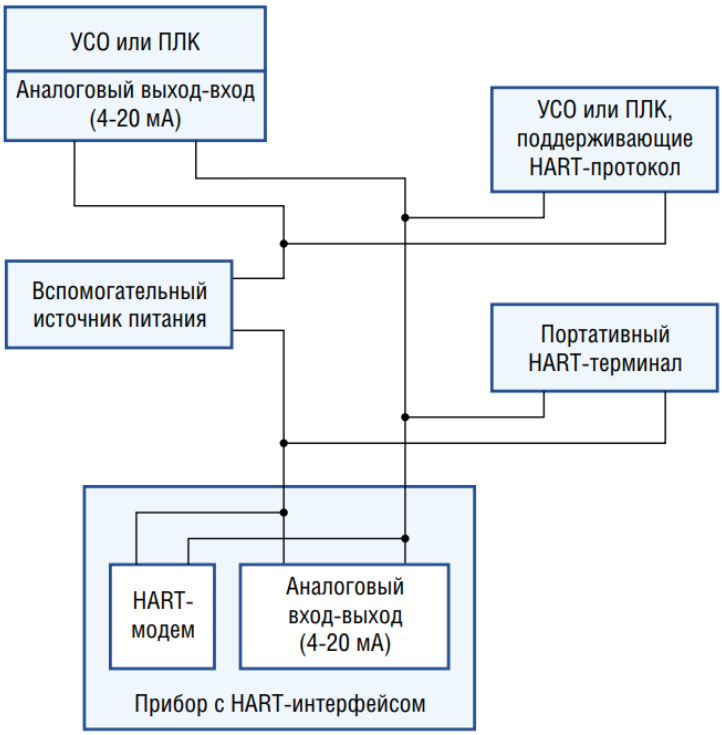

Рисунок 1.13 – Структурная схема подключения HART-устройств. Цифровой канал «точка-точка» с аналоговым сигналом

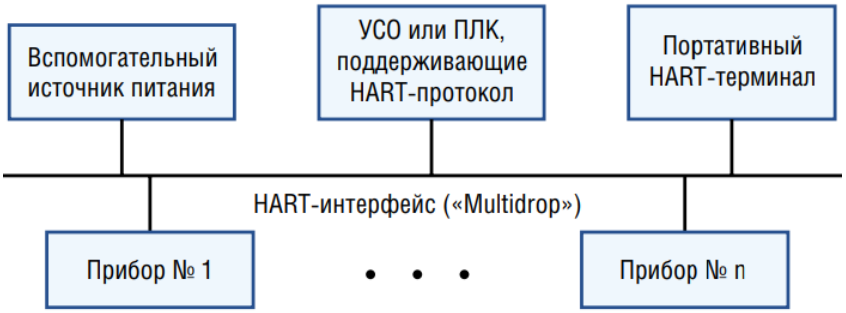

Рисунок 1.14 – Структурная схема подключения HART-устройств. Цифровой канал (топология-шина) без передачи аналогового сигнала

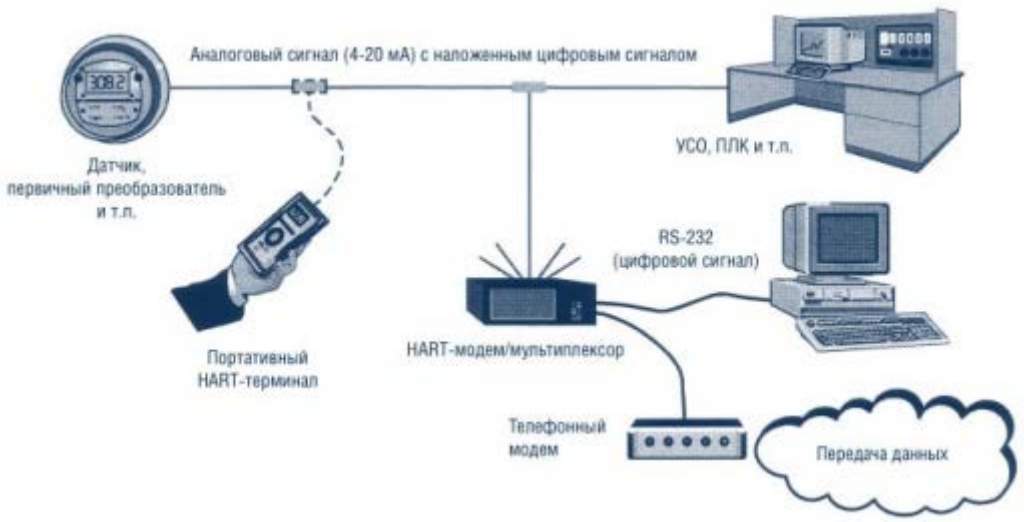

Рисунок 1.15 – Типовые HART-компоненты и схема их подключения

HART-протокол основан на методе передачи данных с помощью частотной модуляции (Frequency Shift Keying, FSK), в соответствии с широко распространенным коммуникационным стандартом Bell 202 [19]. Цифровая информация передается частотами 1200 Гц (логическая 1) и 2200 Гц (логический 0), которые накладываются на аналоговый токовый сигнал (рисунок 1.16). Частотно-модулированный сигнал является двухполярным и при применении соответствующей фильтрации не влияет на основной аналоговый сигнал 4-20 мА. Скорость передачи данных для HART составляет 1,2 кбит/с. Каждый HART-компонент требует для цифровой передачи соответствующего модема.

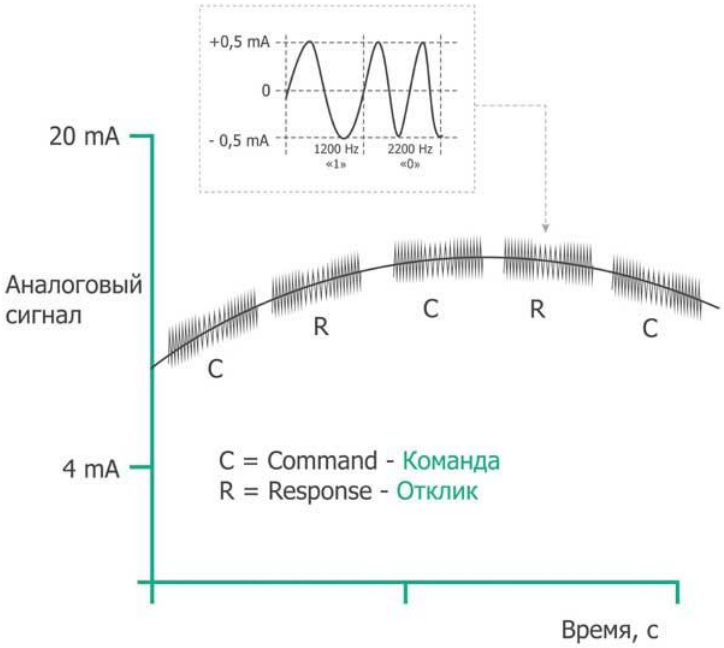

Рисунок 1.16 – Наложение высокочастотных синусоидальных сигналов на аналоговый

Благодаря наличию двух ведущих устройств каждое из них может быть готово к передаче через 270 мс (время ожидания). Цикл обновления данных повторяется 2-3 раза в секунду в режиме запрос/ответ и 3-4 раза в секунду в пакетном режиме. Несмотря на относительно большую длительность цикла, в большинстве случаев он является достаточным для управления непрерывными процессами.

Важнейшим условием для передачи HART-сигналов является то, что нагрузка в общей цепи коммуникационного канала должна быть в пределах 230...1100 Ом. В противном случае возникает несоответствие допустимым значениям параметров сигнала (таблица 2). Нагрузка в основном определяется омической составляющей входного импеданса управляющей системы или портативного HART-терминала. Наряду с омической следует принимать во внимание индуктивную и емкостную составляющие входного импеданса как самого ведущего устройства, так и используемого кабеля, поскольку из-за них происходят задержки и затухание частотной составляющей HART-сигнала.

Таблица 2 – Предельные величины параметров HART-сигналов

Тип устройства	Минимальное значение	Максимальное значение
Ведущее устройство/ HART-терминал: передача сигнала	400 мВ	600 мВ
HART-прибор: передача сигнала	0,8 мА (при нагрузке цепи 230 Ом соответствует 184 мВ)	1,2 мА (при нагрузке цепи 1100 Ом соответствует 1320 мВ)
Ведущее устройство/ HART-терминал, HART-прибор: прием сигнала	Допускается 120 мВ, сигналы менее 80 мВ игнорируются	Допускается 2 В

Для предотвращения недопустимого уровня искажения HART-сигнала максимальное ослабление сигнала HART-прибора должно быть не более 3 дБ на частоте 2500 Гц. Определение этой величины исходит из минимального допустимого значения сигнала, приведенного в таблице 2. Далее устанавливается системная константа, значение которой должно быть менее 65 мкс. В простейшем случае она определяется последовательным сопротивлением и параллельной емкостью цепи. Приблизительный расчет максимальной длины линии связи при условии использования данной константы, а также с учетом влияния параметров кабеля (таблица 3) можно выполнить по следующей формуле:

$$L_{max} = \frac{65 \times 10^{-6}}{RC} - \frac{C_f + 10000}{C}, \quad (1)$$

где

L_{max} — максимальная длина кабеля, м;

R — сопротивление (нагрузка, сопротивление кабеля, внутреннее сопротивление Ex-барьера), Ом;

C — погонная ёмкость кабеля, пФ/м;

C_f — максимальная внутренняя емкость HART-прибора, пФ [19].

Таблица 3 – Типовые параметры кабелей

AWG/сечение, мм²	Погонное сопротивление кабеля, Ом/км	Погонная ёмкость кабеля, пФ/м
14 AWG/2,09	18	150-200
18 AWG/0,8	46	300-420
24 AWG/0,2	178	75-100

Для передачи HART-сигнала могут применяться любые двухпроводные кабели. В зависимости от их исполнения и параметров допускается различная длина линии. Представленные в таблице 4 значения длины линий соответствуют обычному соединению «точка-точка». При многоточечном режиме работы допустимая длина линий будет значительно меньше (таблица 5). Это объясняется, в первую очередь, тем, что в данном случае суммируется параллельная емкость всех подключенных HART-приборов.

Таблица 4 – Типовая длина линий передачи HART-сигнала в зависимости от типа кабеля при соединении «точка-точка»

Исполнение двухпроводной линии	Неэкранированная	Экранированная, витая, многожильный проводник	Экранированная, витая, одножильный проводник
Сечение проводника	24 AWG/0,2 кв. мм	24 AWG/0,2 кв. мм	20 AWG/0,5 кв. мм
Длина линии	«Короткая линия» (длина ограничена внешними факторами)	До 1500 м	До 3000 м

Таблица 5 – Типовая длина линий передачи HART-сигнала в зависимости от погонной емкости кабеля и топологии соединений

Топология соединений	Погонная емкость кабеля		
	400 пФ/м	200 пФ/м	100 пФ/м
«Точка-точка»	600 м	1100 м	2000 м
Многоточечное: 10 HART-приборов, общая ёмкость 5000 пФ	500 м	900 м	1600 м
Многоточечное: 10 HART-приборов, общая ёмкость 22000 пФ	85 м	150 м	250 м

HART-протокол реализует уровни 1, 2 и 7 эталонной модели ISO/OSI-стандарта (таблица 6). Дополнительно протокол предусматривает надстройку к уровню 7 в форме HART Device Description Language.

Таблица 6 – Уровни HART-протокола

Уровень 8 пользовательский	Device Description Language (DDL), EDDL
Уровень 7 прикладной	HART-команды
Уровень 2 канальный	HART-протокол Структура пакета
Уровень 1 физический	Стандарт Bell 202

При реализации уровня 1 HART-протокол опирается на хорошо известный стандарт Bell 202. Таким образом, аппаратно он ориентирован на так называемые Bell- или HART-модемы. На уровне 2 реализуется протокол передачи данных, который использует принцип «ведущий-ведомый» (master-slave). Ведущими могут быть, например, портативный HART-терминал или ПЛК. Активное ведущее устройство передает соответствующую HART-команду на ведомое HART-устройство (как правило, приборы низовой автоматики). Запрашиваемое HART-устройство интерпретирует соответствующую команду и отвечает. Оба ведущих имеют различные адреса, что и гарантирует однозначность при обмене командами и ответами. Передача данных происходит асинхронно в полудуплексном режиме. Структура пакетов во всех режимах работы одинаковая, что создает однозначное соответствие между HART-командами и ответами устройств в многоточечном режиме. Различия существуют только между структурой запроса ведущего (HART-терминала или ПЛК, рисунок 1.17) и структурой ответа ведомых устройств (рисунок 1.18).

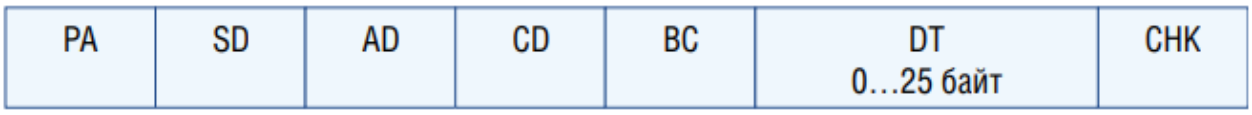

Рисунок 1.17 – Структура HART-телеграммы (запрос от ведущего устройства)

PA	SD	AD	CD	BC	ST	DT 0…25 байт	CHK

Рисунок 1.18 – Структура HART-телеграммы (ответ от ведомого устройства)

Все HART-сообщения передаются побайтно. Байт данных содержит стартовые и стоповые биты, а также бит паритета (рисунок 1.19).

Стартовый бит 0	D7	D6	D5	D4	D3	D2	D1	D0	Бит паритета	Стоповый бит 1

Рисунок 1.19 – Структура HART-телеграммы (формат байта)

Надежность передачи данных по HART-протоколу обеспечивается различными мерами контроля как на уровне байта, так и на уровне пакета. Частота возникновения ошибки на уровне передачи битов составляет 1 ошибку на 105 бит. Каждый передаваемый байт внутри HART-пакета имеет бит паритета; каждый HART-пакет имеет контрольную сумму, с помощью которой можно распознавать до 3 ошибочных битов. Внутри уровня 7 протокол HART использует команды, которые подразделяются на три основных класса (таблица 7).

1) Универсальные команды. Эти команды используются и поддерживаются всеми ведомыми приборами. Они служат решению таких общих задач, как, например, считывание первичных значений измерений, диапазона измерений, граничных величин или констант. Имеется 10 таких команд;

2) Стандартные команды. Они используются в большинстве HART-приборов, но не во всех. К этой группе принадлежат прежде всего такие команды, как считывание и запись стандартных и приборных параметров (например, установить фиксированное значение выходного тока);

3) Специфические команды устройств. Эти команды содержат функции, которые ограничиваются данной моделью или типом прибора. К ним относятся команды, связанные с настройкой, вводом в эксплуатацию или

работой специфических приборов (например, калибровка ультразвукового датчика или считывание базовых данных прибора) [20].

Таблица 7 – Команды HART

Универсальные	**Стандартные**	**Специфические команды устройств**
Read manufacturer and device type	Read selection of up to four dynamic variables	Read or write low&flow cut-off
Read primary variable (PV) and units	Write damping time constant	Start, stop, or clear totalized
Read current output and percent of range	Write device range values	Read or write density calibration factor
Read up to four predefined dynamic variables	Calibrate (set zero, set span)	Choose PV (mass, flow or density)
Read or write 8-character tag, 16-character descriptor, date	Set fixed output current	Read or write materials or construction information
Read or write 32-character message	Perform self-test	Trim sensor calibration
Read device range values, units and damping value (time constant)	Perform master reset	PID enable
Read sensor serial number and limits	Trim PV zero	Write PID set point
Read or write final assembly number	Write PV units	Valve characterization
Write polling address	Trim DAC zero and gain	Valve set point
	Write transfer function (square root or linear)	Travel limits
	Write sensor serial number	User units
	Read or write dynamic variable assignments	Local display information

1.4.2. CAN

CAN (Controller Area Network) представляет собой комплекс стандартов для построения распределенных промышленных сетей, который использует последовательную передачу данных в реальном времени с очень высокой

степенью надежности и защищенности. Центральное место в CAN занимает протокол канального уровня модели OSI. Первоначально CAN был разработан для автомобильной промышленности, но в настоящее время быстро внедряется в область промышленной автоматизации. Начало развития CAN было положено компанией Bosch в 1983 г., первые микросхемы CAN контроллеров были выпущены фирмами Intel и Philips в 1987 году.

В России интерес к CAN за последние годы сильно возрос, однако контроллерного оборудования для CAN в России крайне мало. CAN охватывает два уровня модели OSI: физический и канальный. Стандарт не предусматривает никакого протокола прикладного (7-го) уровня модели OSI.

CAN характеризуется следующими основными свойствами:

1. Каждому сообщению (а не устройству) устанавливается свой приоритет.
2. Гарантированная величина паузы между двумя актами обмена.
3. Гибкость конфигурирования и возможность модернизации системы.
4. Широковещательный прием сообщений с синхронизацией времени.
5. Непротиворечивость данных на уровне всей системы.
6. Допустимость нескольких ведущих устройств в сети (многомастерная сеть).
7. Способность к обнаружению ошибок и сигнализации об их наличии.
8. Автоматический повтор передачи сообщений, доставленных с ошибкой, сразу, как только сеть станет свободной.
9. Автоматическое различение сбоев и отказов с возможностью автоматического отключения отказавших модулей.

К недостаткам можно отнести сравнительно высокую стоимость CAN-устройств, отсутствие единого протокола прикладного уровня, а также чрезмерную сложность и запутанность протоколов канального и прикладного уровня [21].

1.4.3. Profibus

Profibus (PROcess FIeld BUS) – стандарт автоматизации на базе полевой шины. Посредством одного шинного кабеля PROFIBUS связывает контроллеры или системы управления с децентрализованными полевыми устройствами (датчиками и исполнительными механизмами) на полевом уровне, а также обеспечивает согласованный обмен данными с системами связи более высокого уровня.

Profibus имеет три модификации:

1. Profibus DP (Profibus for Decentralized Peripherals, Profibus для децентрализованной периферии) использует уровни 1 и 2 модели OSI, а также пользовательский интерфейс, который в модель OSI не входит. Profibus DP в отличие от FMS и PA построен таким образом, чтобы обеспечить наиболее быстрый обмен данными с устройствами, подключенными к сети [22].

2. Profibus FMS (Profibus с FMS протоколом) использует уровень 7 модели OSI и применяется для обмена данными с контроллерами и компьютерами на регистровом уровне. Profibus FMS предоставляет большую гибкость при передаче больших объемов данных, но проигрывает протоколу DP в популярности вследствие своей сложности. Profibus FMS и DP используют один и тот же физический уровень, основанный на интерфейсе RS-485, и могут работать в общей сети [23].

3. Profibus PA (Profibus for Process Automation, Profibus для автоматизации технологических процессов) использует физический уровень на основе стандарта IEC 1158-2, который обеспечивает питание сетевых устройств через шину и не совместим с RS-485. Особенностью Profibus PA является возможность работы во взрывоопасной зоне [23].

Не так давно появился стандарт PROFInet, который основан на промышленном интернете. Он способен обеспечивать связь промышленной сети Profibus с офисной сетью Ethernet.

Profibus является многомастерной сетью (с несколькими ведущими устройствами). В качестве ведомых устройств выступают обычно устройства ввода-вывода, клапаны, измерительные преобразователи. Они не могут

самостоятельно получить доступ к шине и только отвечают на запросы ведущего устройства.

1.4.4. Modbus

Modbus – сетевой протокол прикладного уровня, широко используемый в промышленной автоматизации для обмена данными между устройствами

Одним из преимуществ Modbus является отсутствие необходимости в специальных интерфейсных контроллерах (Profibus и CAN требуют для своей реализации заказные микросхемы), простота программной реализации и элегантность принципов функционирования. Все это снижает затраты на освоение стандарта как системными интеграторами, так и разработчиками контроллерного оборудования.

Популярность протокола объясняется, прежде всего, совместимостью с большим количеством оборудования, которое имеет протокол Modbus. Кроме того, Modbus имеет высокую достоверность передачи данных, связанную с применением надежного метода контроля ошибок. Modbus позволяет унифицировать команды обмена благодаря стандартизации номеров (адресов) регистров и функций их чтения-записи.

Основным недостатком Modbus является сетевой обмен по типу «ведущий/ведомый», что не позволяет ведомым устройствам передавать данные по мере их появления и поэтому требует интенсивного опроса ведомых устройств ведущим.

Протокол Modbus имеет два режима передачи: RTU (Remote Terminal Unit, «удаленное терминальное устройство») и ASCII. Стандарт предусматривает, что режим RTU в протоколе Modbus должен присутствовать обязательно, а режим ASCII является опционным. Пользователь может выбирать любой из них, но все модули, включенные в сеть Modbus, должны иметь один и тот же режим передачи.

Стандарт Modbus предусматривает применение физического интерфейса RS-485, RS-422 или RS-232. В стандарте Modbus имеются обязательные требования, рекомендуемые и опционные. Модель OSI протокола Modbus содержит три уровня: физический, канальный и прикладной [24].

1.5. Драйвер (описание) устройств полевой шины

Разнообразие измерительных устройств требует единого, структурированного и масштабируемого подхода к используемому программному обеспечению для их управления. Для этого в 1990 году был разработан инструмент для описания работы электронного прибора – EDDL (Electronic Device Description Language). Он позволяет создать функционал, чтобы настраивать, калибровать и устранять неполадки устройств полевой шины без каких-либо предварительных знаний о них. Работает с различными протоколами, такие как HART, Fieldbus и Profibus.

EDDL позволяет инженерам интегрировать информацию из разных типов устройств, использующих разные протоколы, и поддерживать актуальность программного обеспечения по мере появления новых версий устройств. Язык можно сравнить с HTML (гипертекстовой язык разметки) – базовой технологией, которая позволяет просматривать информацию с веб-страниц в Интернете с любого компьютера. EDDL преобразовывает сигналы в значения, текст и графику для отображения понятных человеку данных конфигурирования и обслуживания. Он упрощает работу техников, позволяя им использовать единый инструмент для всех поддерживаемых ими устройств, что позволяет избегать недостатков, связанных с установкой программного обеспечения для каждого типа датчика.

Так, оборудование, которое повсеместно используется в жизни людей (принтеры, клавиатуры, наушники и пр.), взаимодействующие с операционными системами, имеют уникальные драйверы. Но программные средства управления промышленными устройствами должны работать с

сотнями их типов и множеством версий каждого из них, которых по мере появления новых моделей и версий становится все больше. Поэтому требуется более масштабируемое решение. Используя EDDL, производители устройств описывают всю информацию, содержащуюся в их приборе, и то, как эта информация должна отображаться наиболее оптимальным способом для технического специалиста, исключая написание программного обеспечения для каждого конкретного устройства.

EDDL обладает рядом преимуществ, которые служат для удовлетворения требований производства к устройствам, управлению и долгосрочному обслуживанию:

1. Абсолютная поддержка функциональности. Разработчиками создается программное обеспечение для пошаговой настройки, отображения диагностической информации и помощи техническому специалисту в анализе. Карманные компьютеры могут получить доступ ко всем функциям даже самых сложных приборов. Язык EDDL включает в себя методы калибровки и комплексной настройки, представление осциллограмм кривых, непрерывных графиков трендов, графических манометры и барографы, а также изображения.

2. Последовательное отображение. Поскольку EDDL работает на всех приборах и не требует драйверов для каждого в отдельности, программное обеспечение диспетчера устройств и инструменты для его использования являются последовательными и простыми в использовании. Производители решают, что будет отображаться на экране их датчика, чтобы облегчить пользователю настройку, калибровку и диагностику. При этом гарантируется, что различные устройства от разных производителей отображаются в одном и том же стиле.

3. Аппаратная независимость. Программный код на языке EDDL продолжает работать даже при установке новых версии операционных систем или обновлений, поскольку является стандартным и не является уникальным для каждого прибора. Такая независимость является одним из самых больших

преимуществ поскольку это обеспечивает долгосрочную жизнеспособность программного обеспечения и инструментов для управления устройствами.

4. Универсальность. Технология EDDL способна объединить информацию, передаваемую по различным протоколам, позволяя программному обеспечению поддерживать различные типы устройств. На предприятии могут использоваться Foundation fieldbus для устройств управления технологическими процессами, Profibus для двигателей и HART для приборов функциональной безопасности. EDDL позволяет обеспечить функциональную совместимость между этими протоколами.

5. Простота интеграции. Добавить поддержку нового типа или версии модели в систему на основе EDDL очень просто. Достаточно связать новый файл с кодом с существующим программным обеспечением для управления устройствами. Например, новые беспроводные устройства могут быть легко интегрированы в текущую программную реализацию управления устройствами.

6. Внешний доступ. Системы на основе EDDL транслируют информацию с устройств не только для отображения на локальном экране, но также могут сделать ее доступной для других приложений, например, через OPC-сервер (программное решение для управления объектами автоматизации). Это выгодно отличается от систем, которые передают данные только на экран.

7. Надежность. Системы управления устройствами или карманные компьютеры на основе EDDL не выполняют вызовов процедур стороннего программного обеспечения. Это позволяет избежать проблем, которые могут возникнуть, если драйверы устройств разработаны несколькими производителями. При этом каждая версия имеет уникальный EDDL-файл, снижая риск, связанный со сменой версий программного обеспечения.

8. Удобство тестирования. Поскольку файлы EDDL тестируются вместе с устройством в рамках проверки совместимости протокола HART и полевой шины Foundation, обеспечивается безрисковая интеграция сотен устройств различных типов и версий.

9. Безопасность. При добавлении прибора с файлами EDDL в системы автоматизаций не устанавливается программное обеспечение и не делается никаких записей в реестре, что сводит к минимуму риск появления вредоносного кода. Это обеспечивает более высокий уровень безопасности по сравнению с решениями, основанные на драйверах устройств, которые могут быть заражены.

1.5.1. Обзор существующих разработок драйверов для датчиков давления

На рынке представлены готовые драйверы для датчиков давления от других производителей. Список и характеристики некоторых разработок представлены в таблице 8.

Таблица 8 – Драйверы для датчиков давления

№	Производитель	Датчик	Интерфейс	Технология	Язык	Протокол
1	ООО НПП «Элемер»	АИР-20/М2-Н	Токовая петля	DTM	C#	HART
2	АО «ПГ «Метран»	Метран-75	Токовая петля	DTM	C#	HART
3	АО «ЭМИС»	ЭМИС-БАР	Токовая петля	DD	EDDL	HART
4	ООО «ЭЛЕМЕР-УФА»	АИР-20/М2-Н-10	Токовая петля	DD	EDDL	Modbus

Выводы по разделу один

Исследования рынка промышленных сетей показывают, что использование и внедрение промышленного интернета (65% мирового рынка промышленной

автоматизации) увеличивается с каждым годом. Однако за полевой шиной остается значительная часть – 28% [25]. Это объясняется многими объективными причинами: начиная от экономической нецелесообразности радикальной замены существующих систем, и, заканчивая государственными стандартами, исключающие возможное использование других технологий. К примеру, согласно ГОСТ Р МЭК 61511-1-2018 к таковым относятся приборные системы безопасности для промышленных процессов, которые применяются на нефтехимических и нефтеперерабатывающих производствах [26].

Согласно таблице 8 существуют готовые программные решения для обеспечения работы датчиков. Однако они являются интеллектуальной собственностью сторонних предприятий, поэтому не могут использоваться. В ином случае драйвер основан на других технологиях, не подходящих для использования датчиком «Метран-150».

Датчик «Метран-150», для которого необходимо разработать драйвер, основан на методах емкостной ячейки (применяется при измерении разности давления и небольших значений (меньше 40 кПа) избыточного давления) и тензорезистивного сенсора (используется при измерении абсолютного и избыточного давления с диапазоном от 40 кПа до 68 МПа). Данные передаются по токовой петле 4-20 мА с выделением из нее HART сигнала для передачи цифровой информации.

2. ФОРМУЛИРОВАНИЕ ТРЕБОВАНИЙ, РАЗРАБОТКА АРХИТЕКТУРЫ ДРАЙВЕРА

В данном разделе необходимо сформулировать функциональные и нефункциональные требования и разработать архитектуру, которая включает в себя:

1. Отображение взаимодействия драйвера в средстве интеграции полевых устройств.
2. Пользовательское моделирование.
3. Функциональное моделирование.

2.1. Функциональные требования

1. Совместимость с различными версиями устройств.
2. Поддержка HART-протокола связи.
3. Возможность настройки и калибровки: сброс настроек, ведение журнала всех операций, включение оповещений уведомлений о достижении заданных пороговых значений давления, перезагрузка устройства.
4. Защита данных и безопасность – ограничение возможности изменения настроек.

2.2. Нефункциональные требования

1. Пользовательский интерфейс драйвера должен быть интуитивно понятным и простым в использовании.
2. Драйвер должен поддерживать русский язык.
3. Интерфейс должен корректно работать на различных разрешениях экрана.
4. Драйвер должен предоставлять возможность сохранения и загрузки настроек и калибровочных данных.

2.3. Архитектура

Для интеграции измерительных устройств, в том числе датчиков давления, используется технология FDI (Field Device Integration, интеграция полевых устройств). Она обеспечивает работу приборов вне зависимости от их производителя и версии. Это достигается благодаря FDI-пакетам, который используется всеми системами, включая персональные компьютеры, полевые устройства, системы управления процессами и автоматизации. Это решает проблему интеграции полевых устройств с множеством сетей, операционных систем и систем управления, используемых в обрабатывающей промышленности.

Драйвер устройства является составной частью FDI-пакета. Программный код, написанный на EDDL, в настоящее время не отдельная сущность, а одна из составляющих технологии интеграции полевых устройств FDI, архитектура которой представлена на рисунке 2.1.

Согласно архитектуре, измерительные приборы коммуницируют с FDI-сервером с помощью OPC UA (Open Platform Communications Unified Architecture, кроссплатформенная открытая платформа коммуникации). На сервере имеются FDI-пакеты, описывающие работу устройств, бизнес-логику и описание пользовательского интерфейса. Взаимодействие FDI-клиента с сервером происходит также через OPC UA.

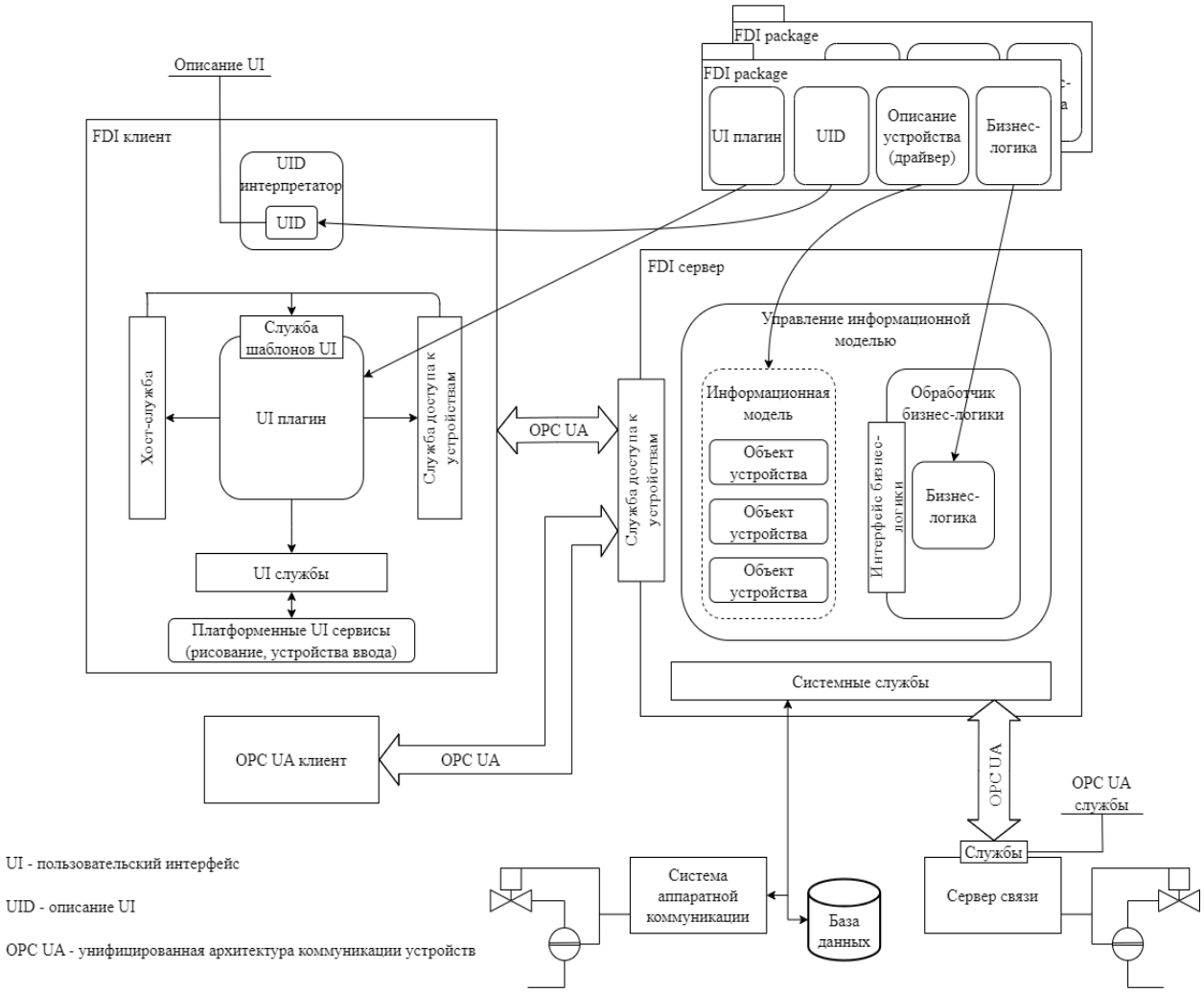

Рисунок 2.1 – Архитектура FDI

Для разработки архитектуры драйвера будет использовать нотация IDEF0. С ее помощью будут отражены потоки данных и основные функциональные возможности. На рисунке 2.8 представлена диаграмма A-0, на которой показан верхний уровень.

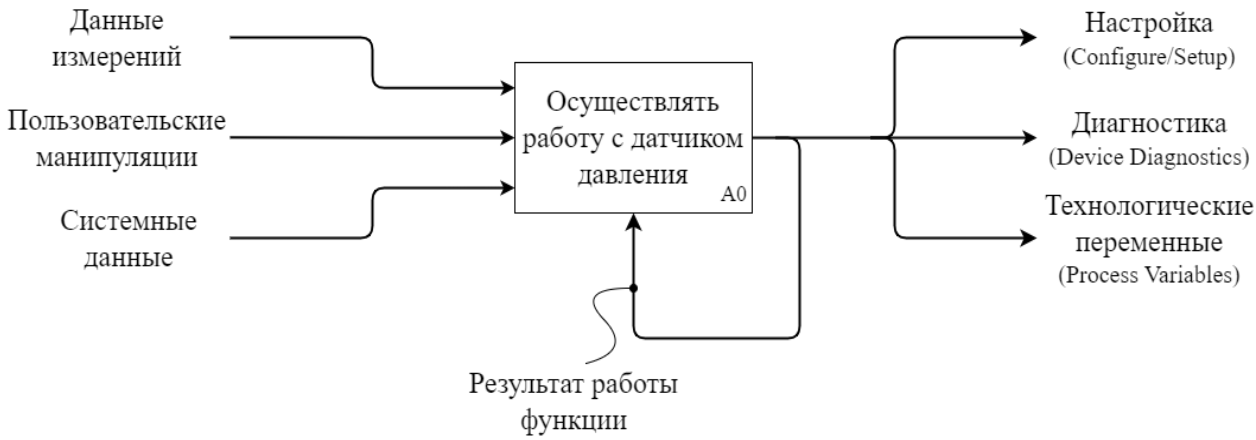

Рисунок 2.8 – Диаграмма A-0

Драйвере устройства содержит три основные функциональные группы: настройка, диагностика и просмотр значений технологических переменных (рисунок 2.9).

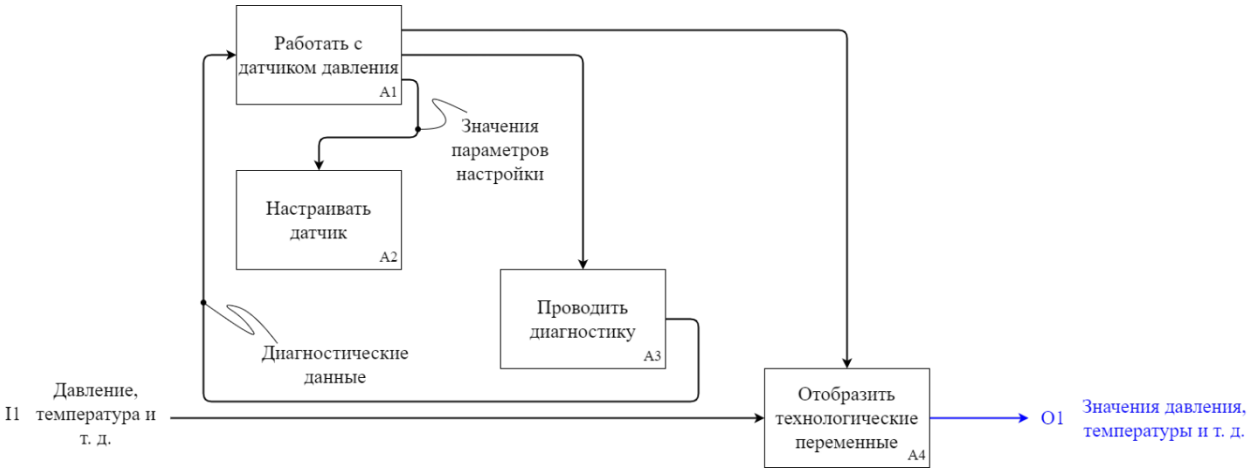

Рисунок 2.9 – Диаграмма с группами функций

На рисунке 2.10 представлена диаграмма с узлом A2 с принципом настройки датчика. В необходимый функционал входят:

1. Калибровка.
2. Ограничение записи – запрет на изменение настроек.
3. Настройка соединения (HART).
4. Настройка индикаторов для сигнализирования о неисправностях.
5. Получение информации об устройстве (датчике).
6. Настройка выхода – форматов измерений.
7. Перезагрузка устройства (датчика).

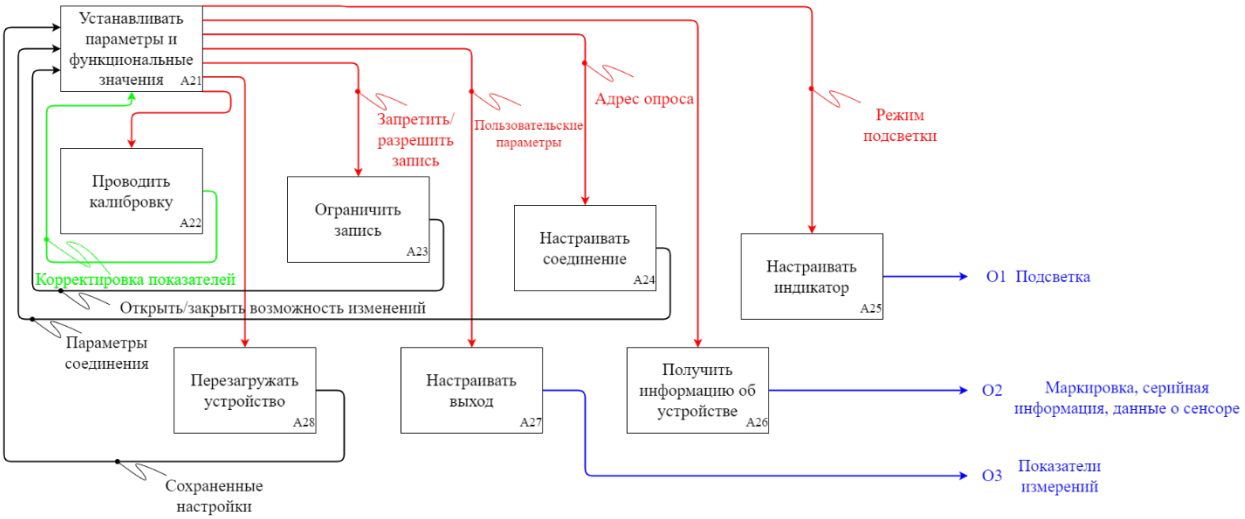

Рисунок 2.10 – Диаграмма узла А2

На рисунке 2.11 представлена диаграмма с узлом А3 с описанием необходимого диагностического функционала. Его главная функция – настройка сигнала тревоги при выходе значений за рамки, установленных техническим специалистом. Также для удобства пользователя необходимо в диагностический раздел добавить функции ограничения записи и получения информации о датчике.

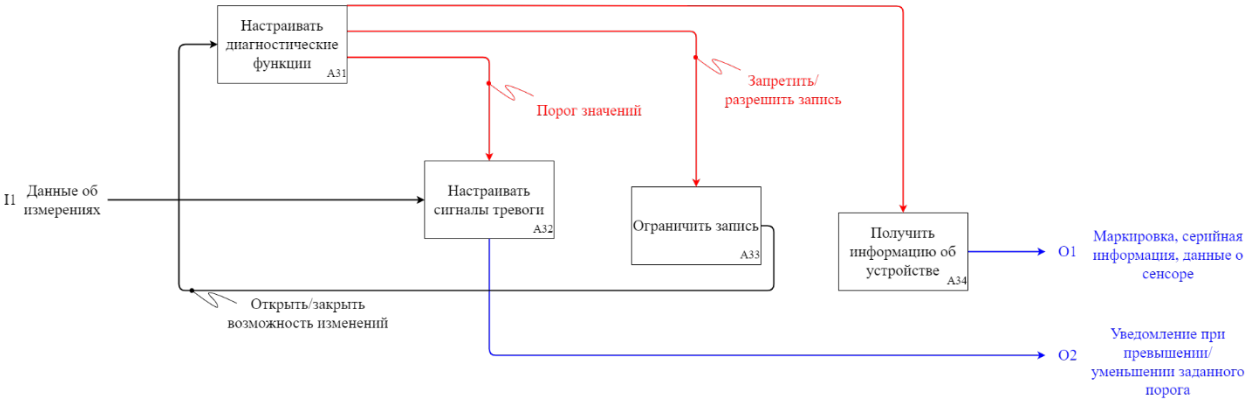

Рисунок 2.11 – Диаграмма узла А3

На рисунке 2.12 представлена диаграмма с узлом А4 с описанием раздела с технологическими переменными. Он содержит в себе две функции: просмотр значений переменных и отображение статуса датчика на основе измеренных данных.

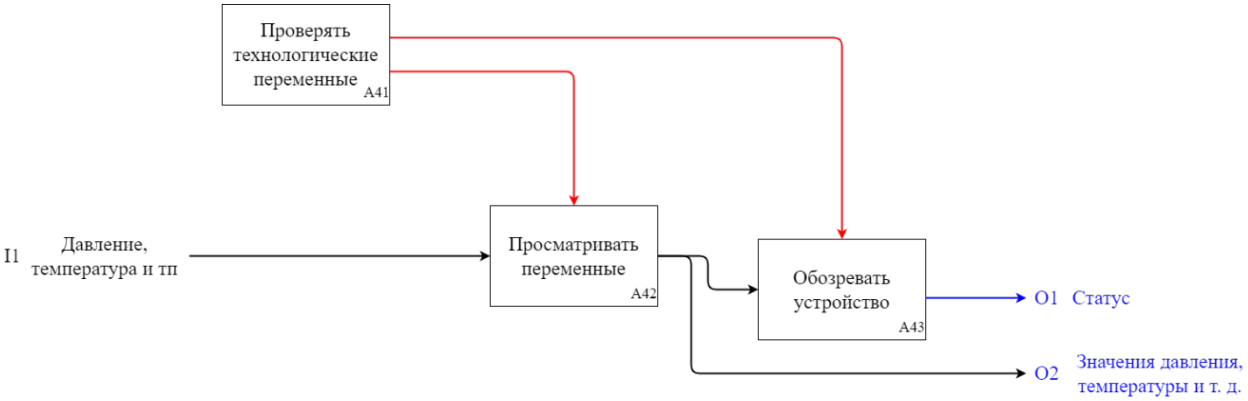

Рисунок 2.12 – Диаграмма узла А4

Выводы по разделу два

В данном разделе сформулированы функциональные и нефункциональные

требования, основными из которых являются: возможность настройки и калибровки измерительного устройства, а также интуитивно понятный пользовательский интерфейс на русском языке.

Помимо этого, разработана архитектура драйвера в виде IDEF0 модели. Описано, как драйвер взаимодействует в системе FDI, а также ее устройство. Представлено описание взаимодействия пользователя, клиента и сервера в формате диаграмм прецедентов (функций).

3. РАЗРАБОТКА И ОТЛАДКА ПРОГРАММНОГО КОДА

Разработка начинается с написания простой функции – перезагрузки устройства. За это отвечает HART-команда под номером 42. Для ее инициирования прибору потребуется до 2 секунд. В течение этого времени аналоговый выход не будет изменяться, и устройство не будет отвечать на запросы HART. Реализация начинается с написания метода, в котором происходит вызов команды (листинг 12).

Листинг 12 – Функция перезагрузки устройства

```
METHOD deviceRestart_method
{
    LABEL deviceRestart_string;
    DEFINITION
    {
        char status[STATUS_SIZE];
        SEND_AND_FAIL_IF_ERROR(42)
        DELAY(6, deviceBeingRestarted_string);
        process_abort();
    }
}
```

Команда отправляется с помощью кода, представленном в листинге 13.

Листинг 13 – Отправка HART-команды

```
#define SEND_AND_FAIL_IF_ERROR(commandNumber)\
    send(commandNumber, status);\
    if(status[STATUS_RESPONSE_CODE] != 0)\
    {\
        display_response_status(commandNumber,
status[STATUS_RESPONSE_CODE]);\
        process_abort();\
    }
```

Чтобы появилась возможность использовать функцию перезагрузки, необходимо создать меню и добавить в него реализованный метод (листинг 14).

Листинг 14 – Создание основного меню и окна с функцией перезагрузки

```
MENU restoreRestartRestoreRestart_page
{
    LABEL restoreRestart_string;
    STYLE PAGE;
    ITEMS
    {
        deviceRestart_method,
        COLUMNBREAK,
        restoreRestartRestoreRestart_string
    }
```

```
}

MENU restoreRestart_window
{
   LABEL restoreRestart_string;
   STYLE WINDOW;
   ITEMS
   {
      restoreRestartRestoreRestart_page
   }
}

MENU device_root_menu
{
   LABEL deviceSettings_string;
   STYLE MENU;
   ITEMS
   {
      restoreRestart_window
   }
}
```

Все использованные строки в листингах 12-14 представлены в листинге 15.

Листинг 15 – Строки для меню и функции перезагрузки

```
#define                                                deviceSettings_string
   “Device Settings”\

   “|ru|Параметры устройства”

#define                                                restoreRestart_string
   “Restore/Restart”\

   “|ru|Восстановление/Перезагрузка”

#define                                                 deviceRestart_string
   “Device Restart”\

         “|ru|Перезапуск устройства”

#define                                                   deviceRestart_help
   “Device Restart- Resets power to the device
                                                (device    configuration    is
                     preserved).”\

   “|ru|Перезапуск устройства-
                                                перезагружает устройство
                                                (конфигурация    датчика    не
                                             изменится).»

   #define deviceBeingRestarted_string                                    “The
device is being restarted. Please wait.»\

   «|ru|Устройство находится в процессе
                                                перезагрузки.      Пожалуйста,
                                             подождите.»
```

При работе с датчиком при помощи коммуникатора, будет отображаться

кнопка для перезагрузки устройства (рисунок 3.1).

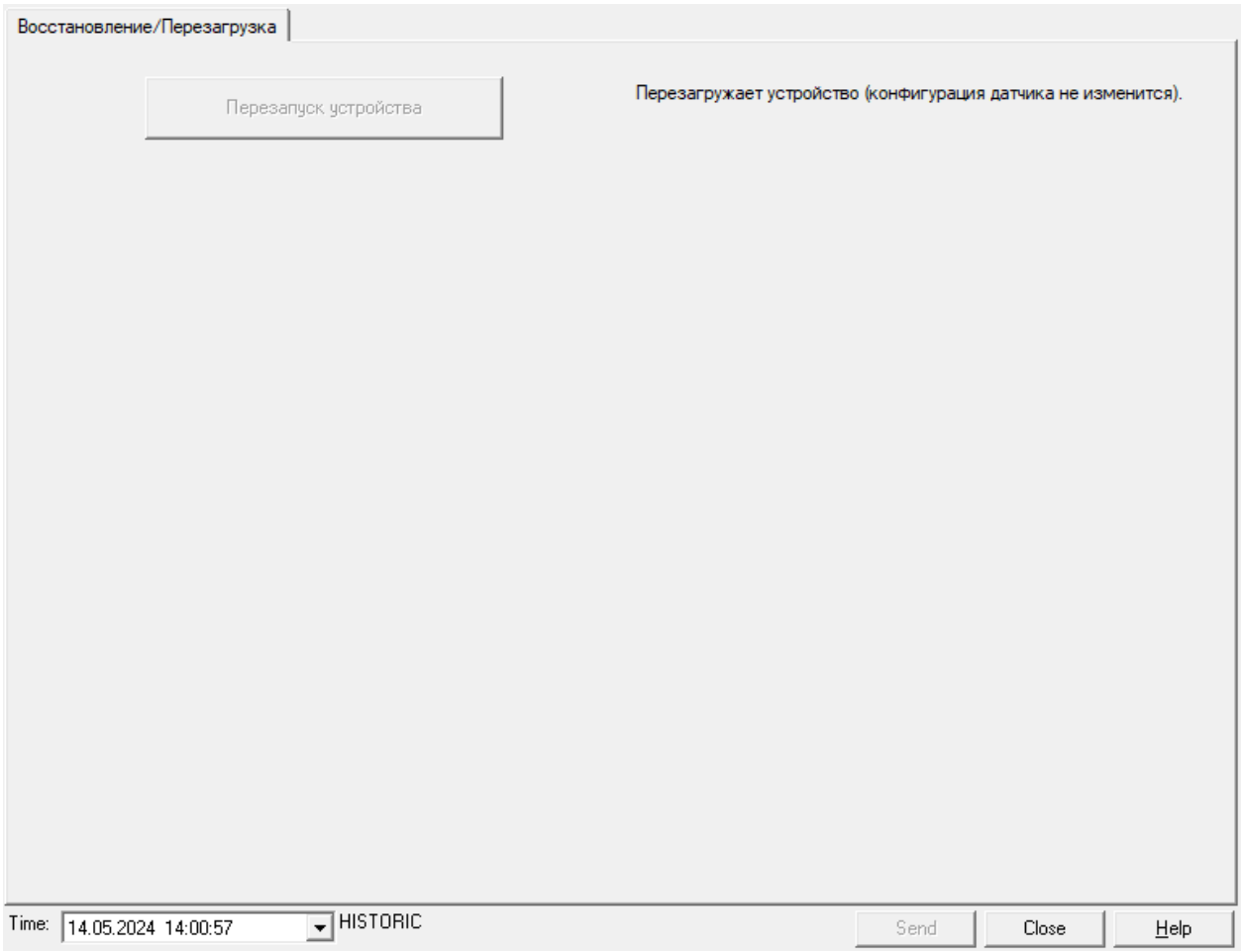

Рисунок 3.1 – Экранная форма перезагрузки устройства

На рисунке 3.2 представлено окно с обзором настроек датчика. В нем есть возможность просмотреть и очистить журнал переменных, а также настроить значения сигналов аварий и насыщения.

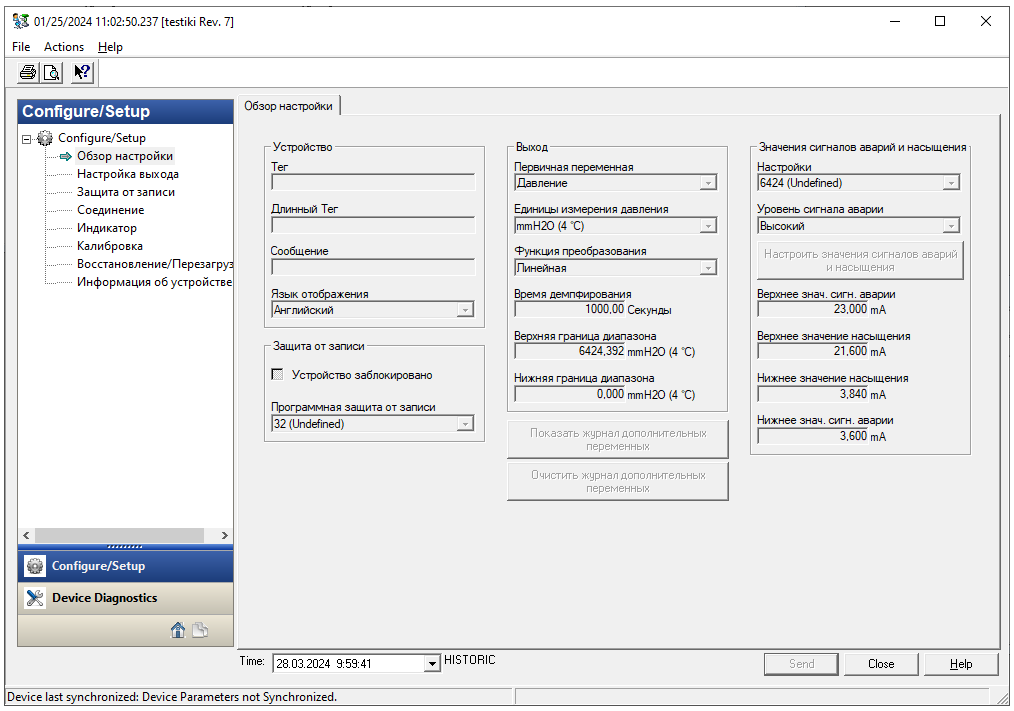

Рисунок 3.2 – Экранная форма обзора настроек

На рисунках 3.3-3.13 представлены окна с настройкой выхода. Во вкладке «Аналоговый выход» (рисунок 3.3) отображаются данные, а также предоставляется возможность настроить первичную переменную, значения сигналов аварий и насыщения.

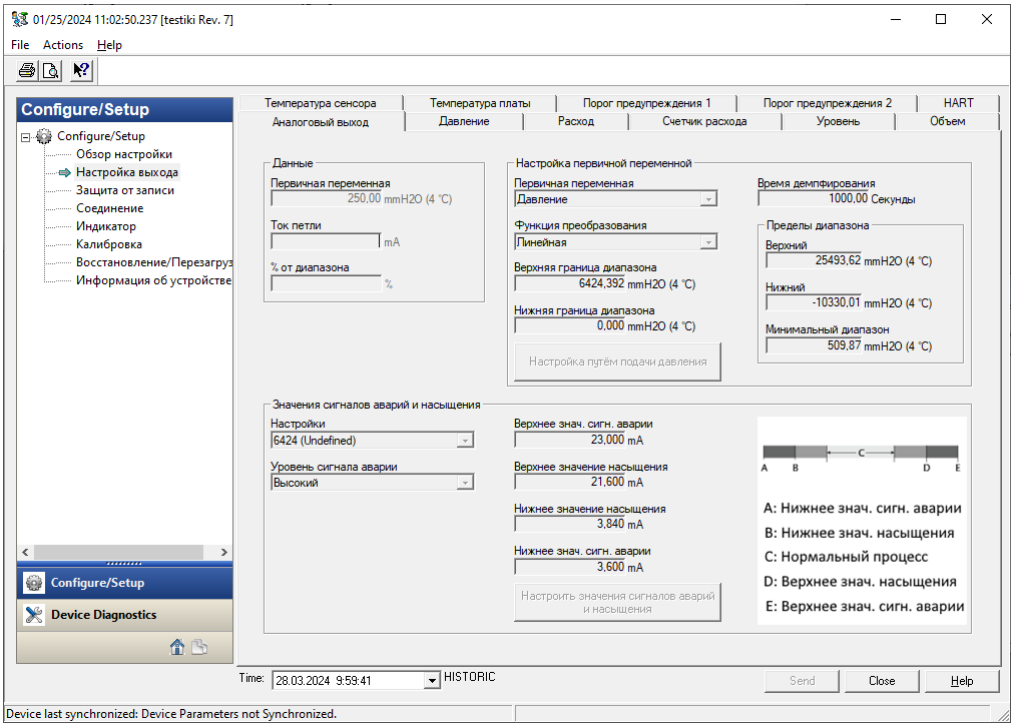

Рисунок 3.3 – Экранная форма настройки аналогового выхода

На рисунке 3.4 отображена вкладка «Давление», на которой находятся данные о давлении, а также предоставляется возможность очистить журнал.

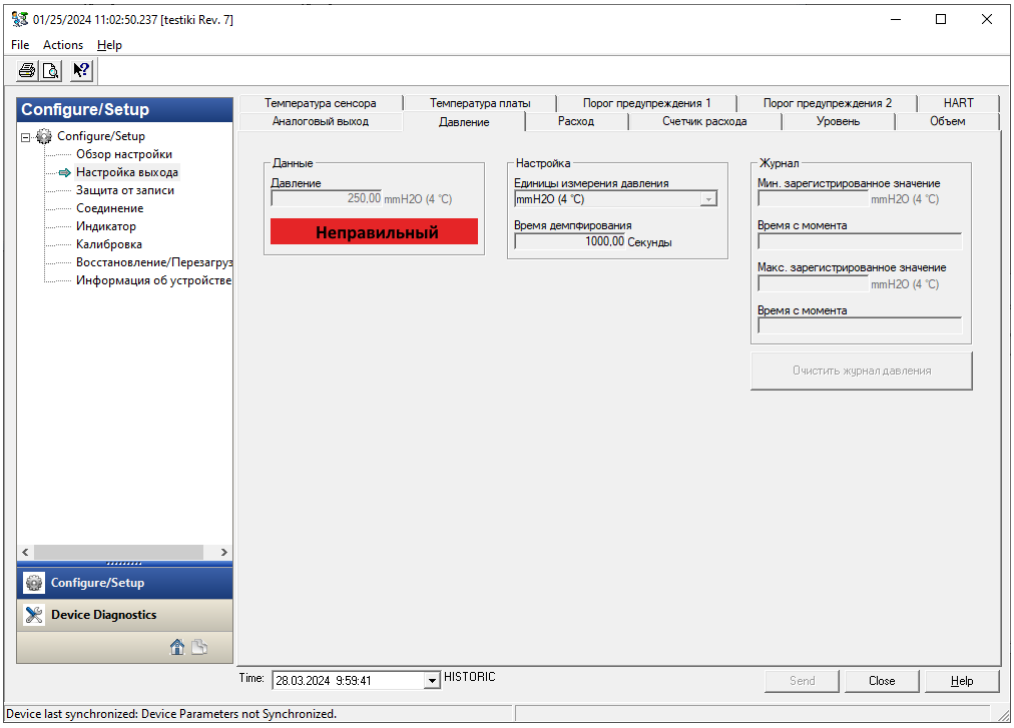

Рисунок 3.4 – Экранная форма настройки выхода (давление)

На рисунке 3.5 отображена вкладка «Расход», на которой находятся данные о расходе, а также предоставляется возможность настроить параметры.

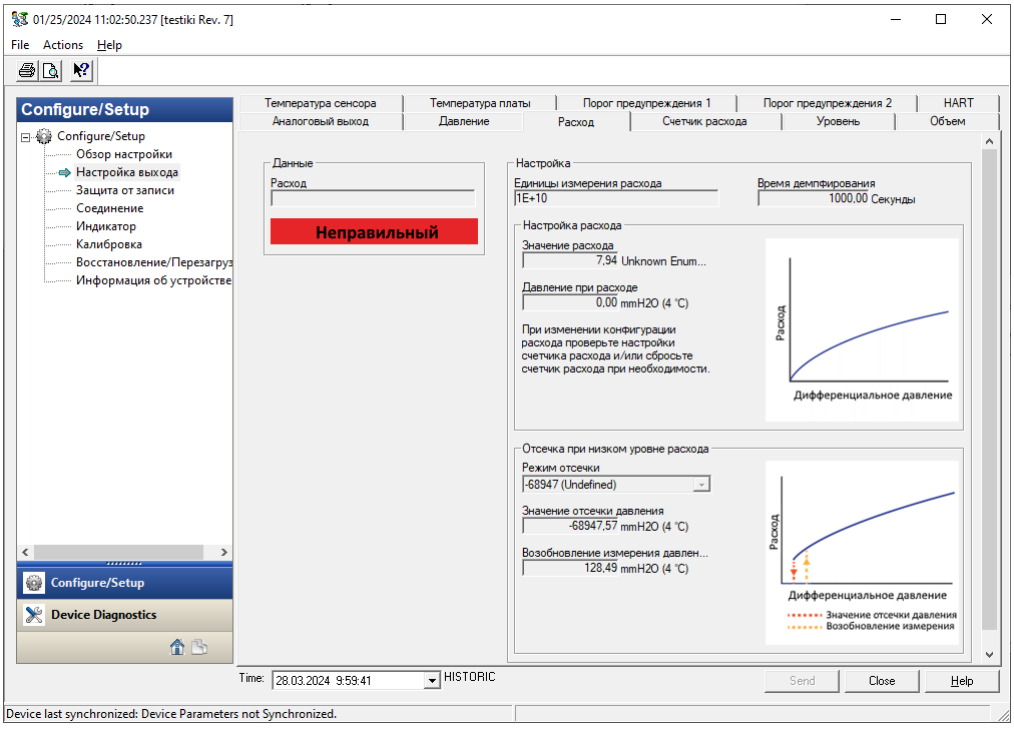

Рисунок 3.5 – Экранная форма настройки выхода (расход)

На рисунке 3.6 отображена вкладка «Счетчик расхода», на которой находятся данные о расходе, а также предоставляется возможность настроить единицы измерения и направление.

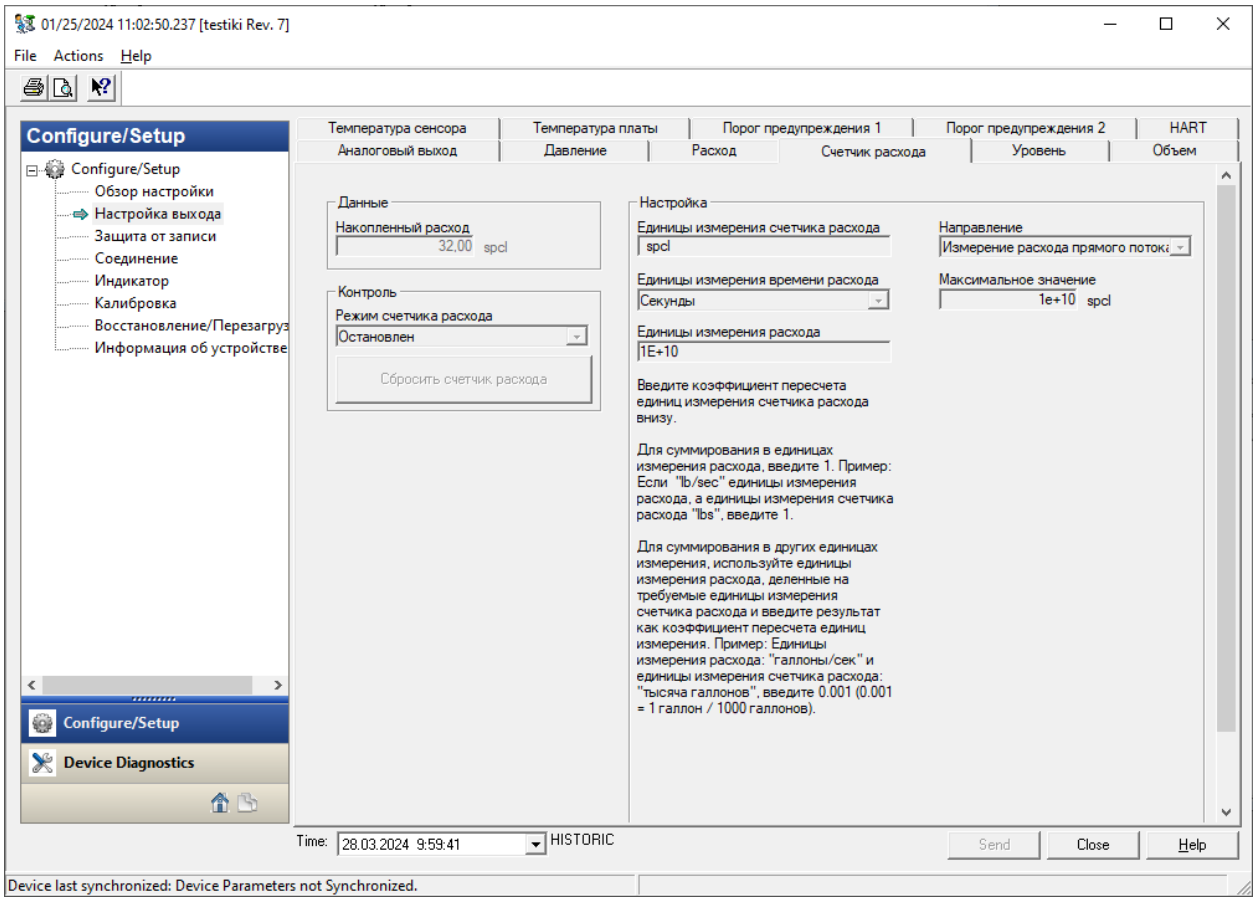

Рисунок 3.6 – Экранная форма настройки выхода (счетчик расхода)

На рисунке 3.7 отображена вкладка «Уровень», на которой находятся данные об уровне, а также предоставляется возможность:

1. Установить пределы.
2. Отрегулировать, сбросить и настроить уровень.
3. Указать единицы измерения.

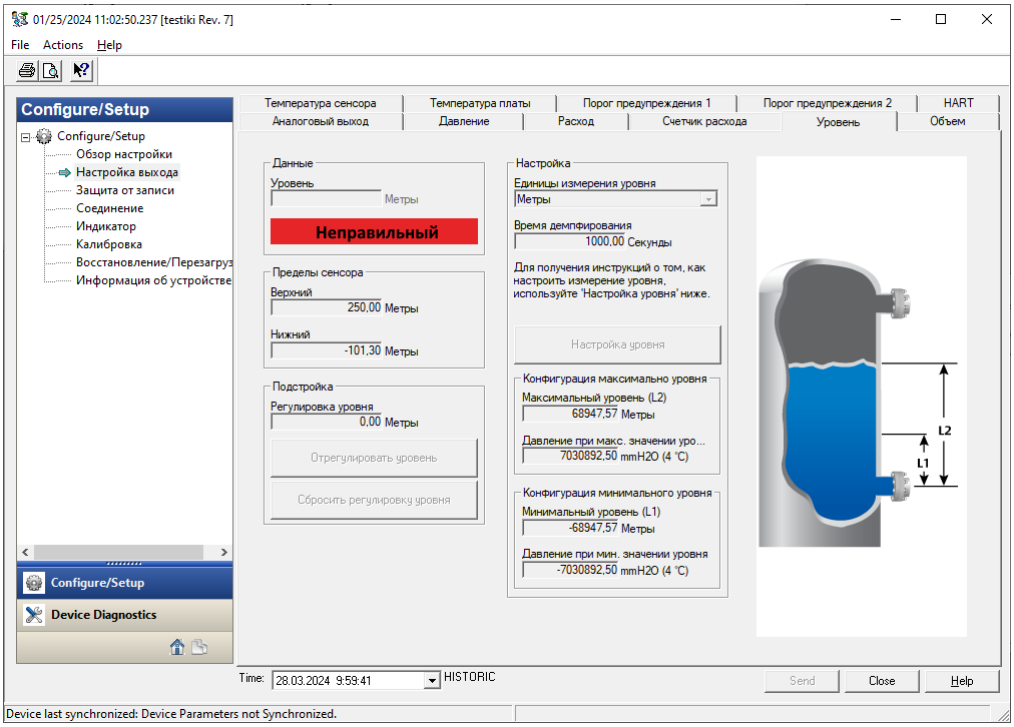

Рисунок 3.7 – Экранная форма настройки выхода (уровень)

На рисунке 3.8 отображена вкладка «Объем», на которой находятся данные об объеме, а также предоставляется возможность сконфигурировать резервуар.

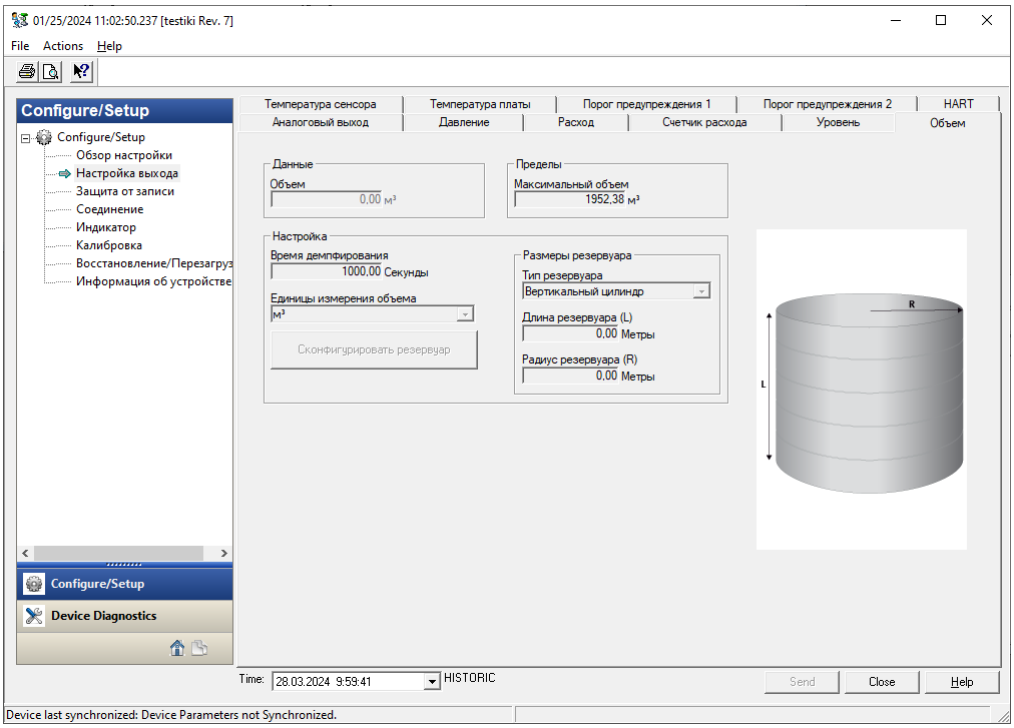

Рисунок 3.8 – Экранная форма настройки выхода (объем)

На рисунке 3.9 отображена вкладка «Температура сенсора», на которой

находятся данные о температуре, а также предоставляется возможность очистить журнал.

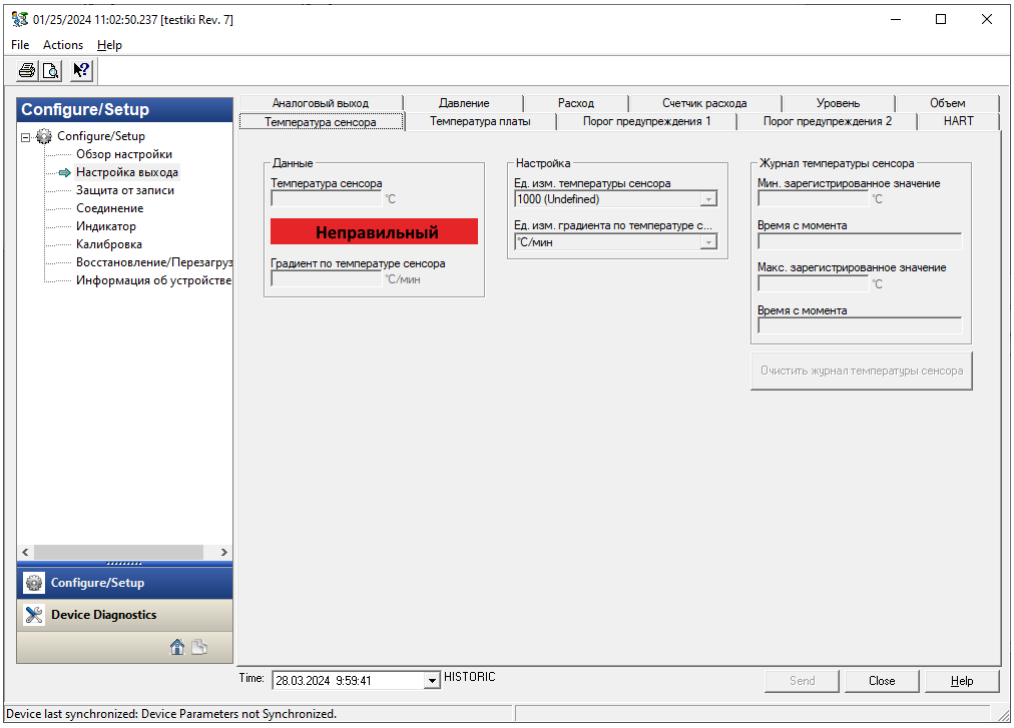

Рисунок 3.9 – Экранная форма настройки выхода (температура сенсора)

На рисунке 3.10 отображена вкладка «Температура платы», на которой находятся данные о температуре, а также функция очистки журнала.

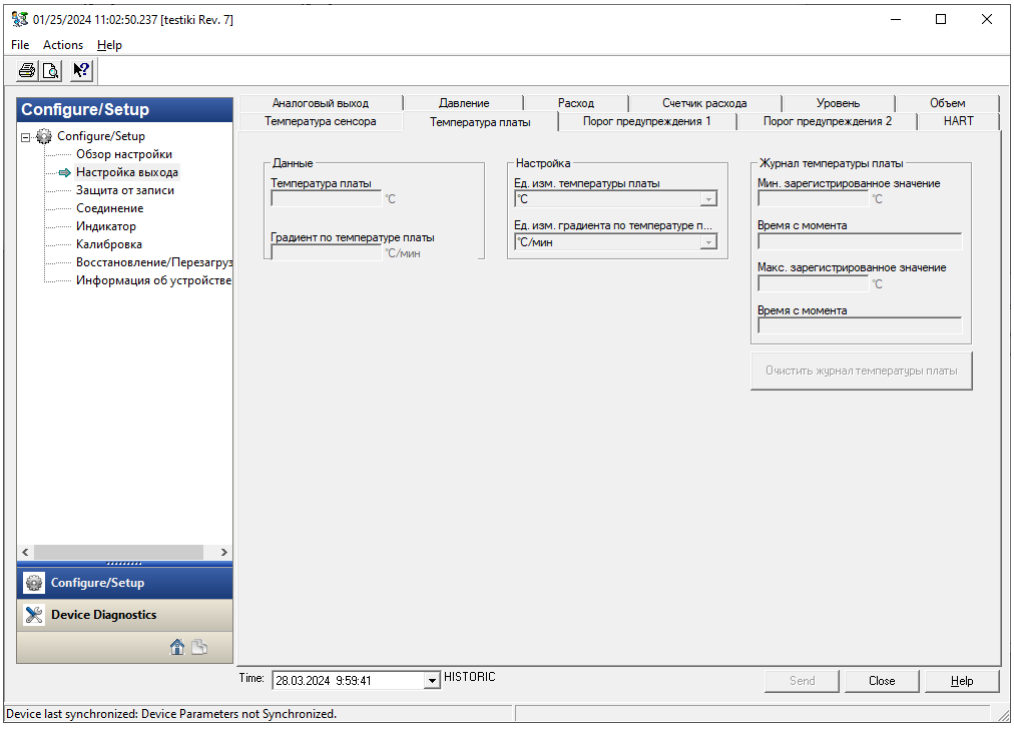

Рисунок 3.10 – Экранная форма настройки выхода (температура платы)

На рисунках 3.11, 3.12 отображены вкладки с настройкой уведомлений при выходе за установленные пользователем границы. На них есть возможность выбрать, какую переменную отслеживать (давление, температуру и т. д.) и установить порог значений. Также имеется график, на котором отображается отслеживаемое значение и границы.

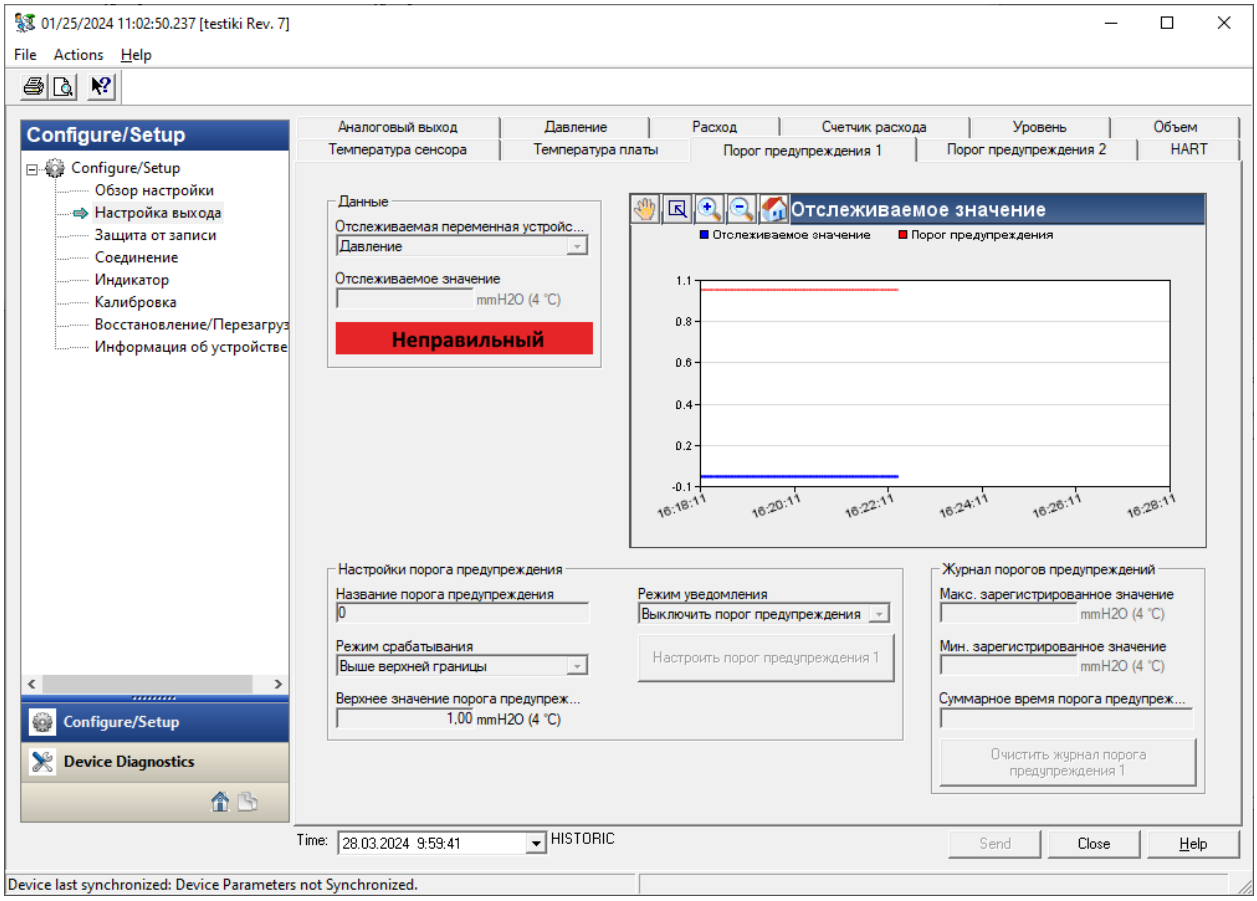

Рисунок 3.11 – Экранная форма настройки выхода (порог предупреждения 1)

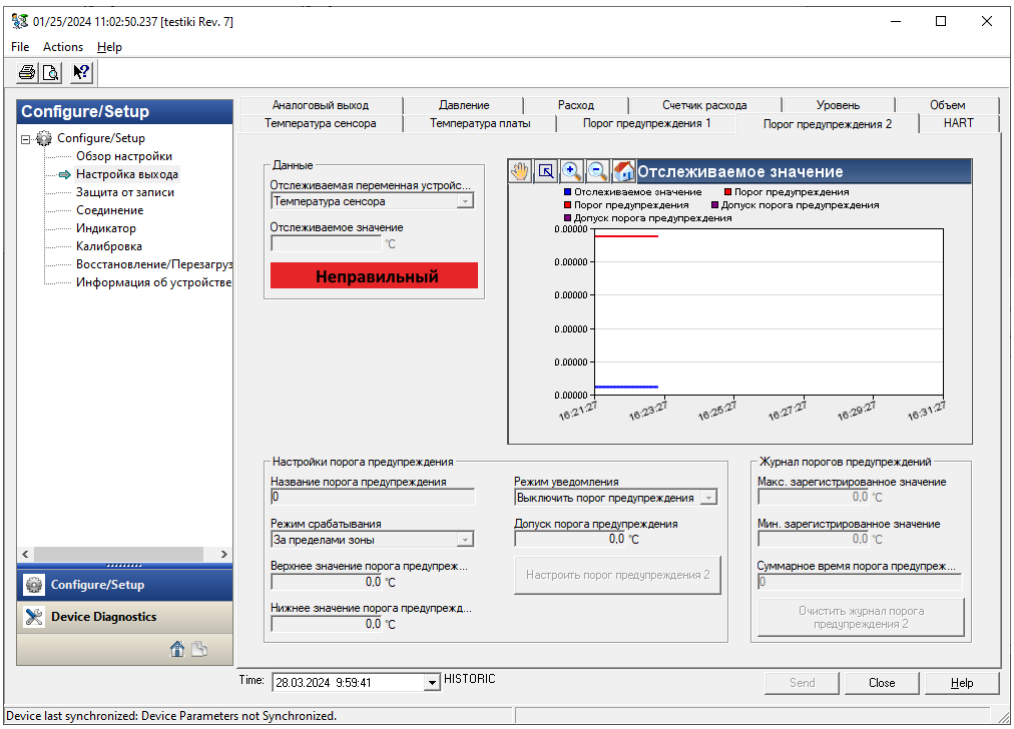

Рисунок 3.12 – Экранная форма настройки выхода (порог предупреждения 2)

На рисунке 3.13 отображена вкладка «HART». На ней редактируются соответствия переменных, а также настраивается соединение.

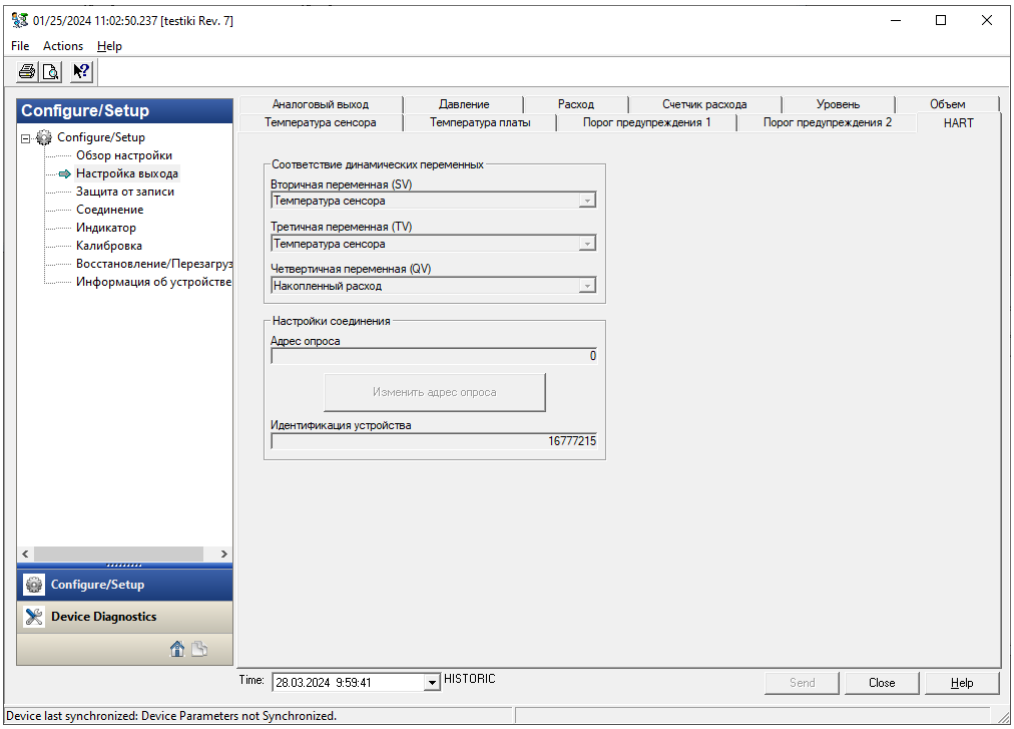

Рисунок 3.13 – Экранная форма настройки выхода (HART)

На рисунке 3.14 отображено окно «Защита от записи». В нем устанавливается или снимается ограничение на конфигурацию и настройку

устройства. Также, имеется возможность задать пароль.

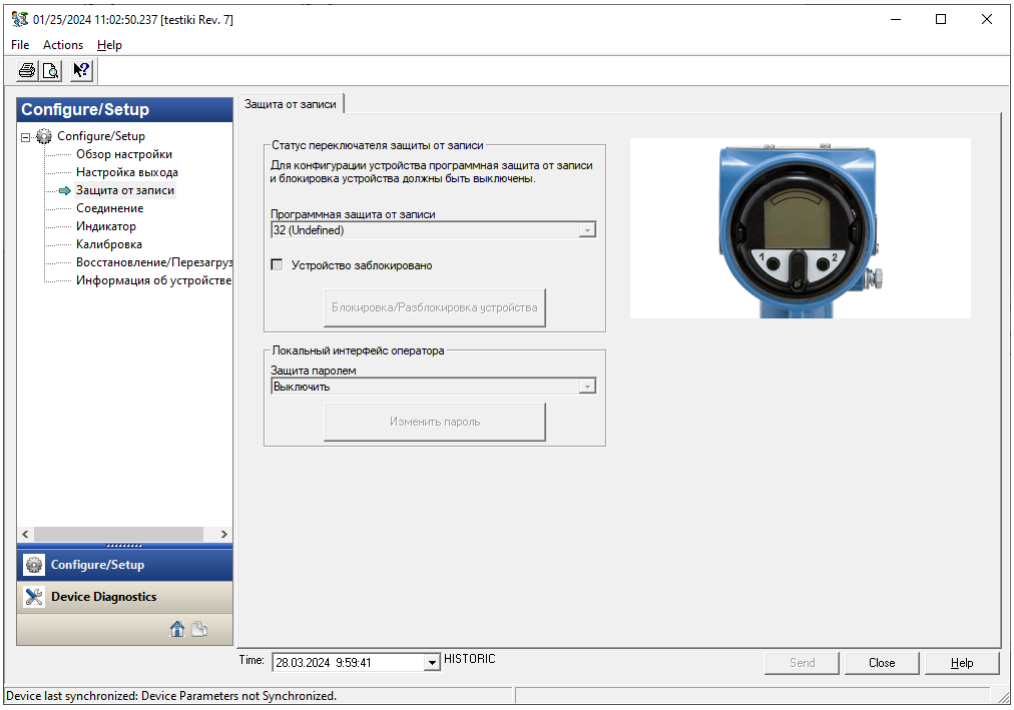

Рисунок 3.14 – Экранная форма настройки защиты от записи

На рисунке 3.15 отображено окно «Индикатор». В нем устанавливаются параметры индикатора и настройки подсветки.

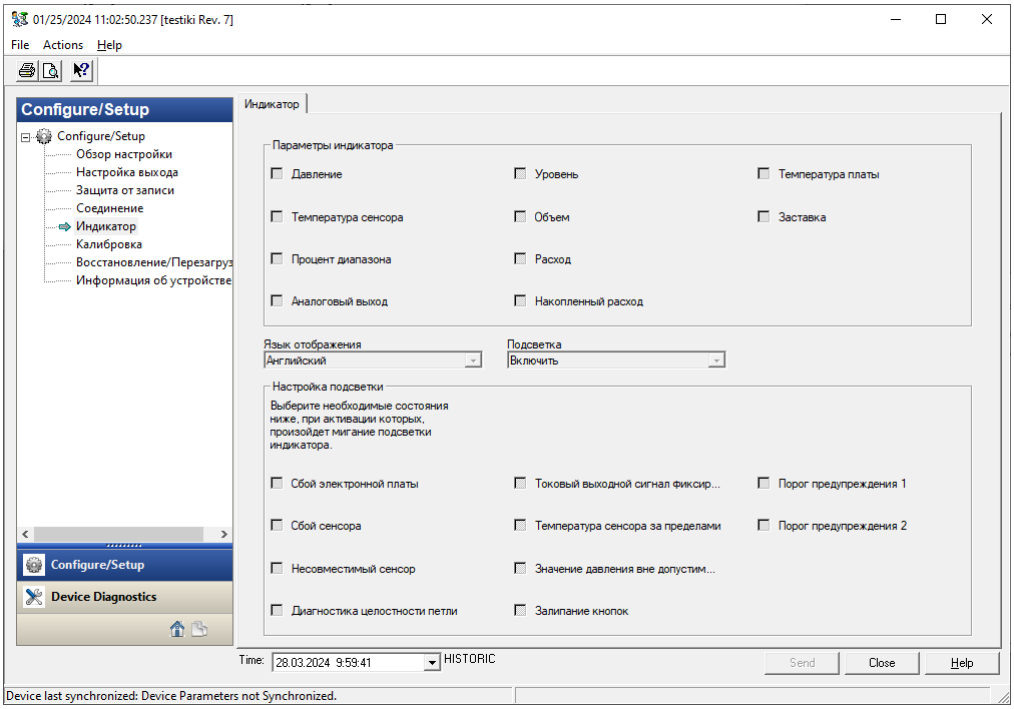

Рисунок 3.15 – Экранная форма настройки индикатора

Одним из самых важных инструментов является калибровка. На

рисунке 3.16 представлено окно с калибровочными функциями давления, к ним относятся:

1. Калибровка нуля.
2. Калибровка нижней и верхней точки сенсора.
3. Показать и очистить журнал калибровки.
4. Восстановить заводские настройки калибровки давления.

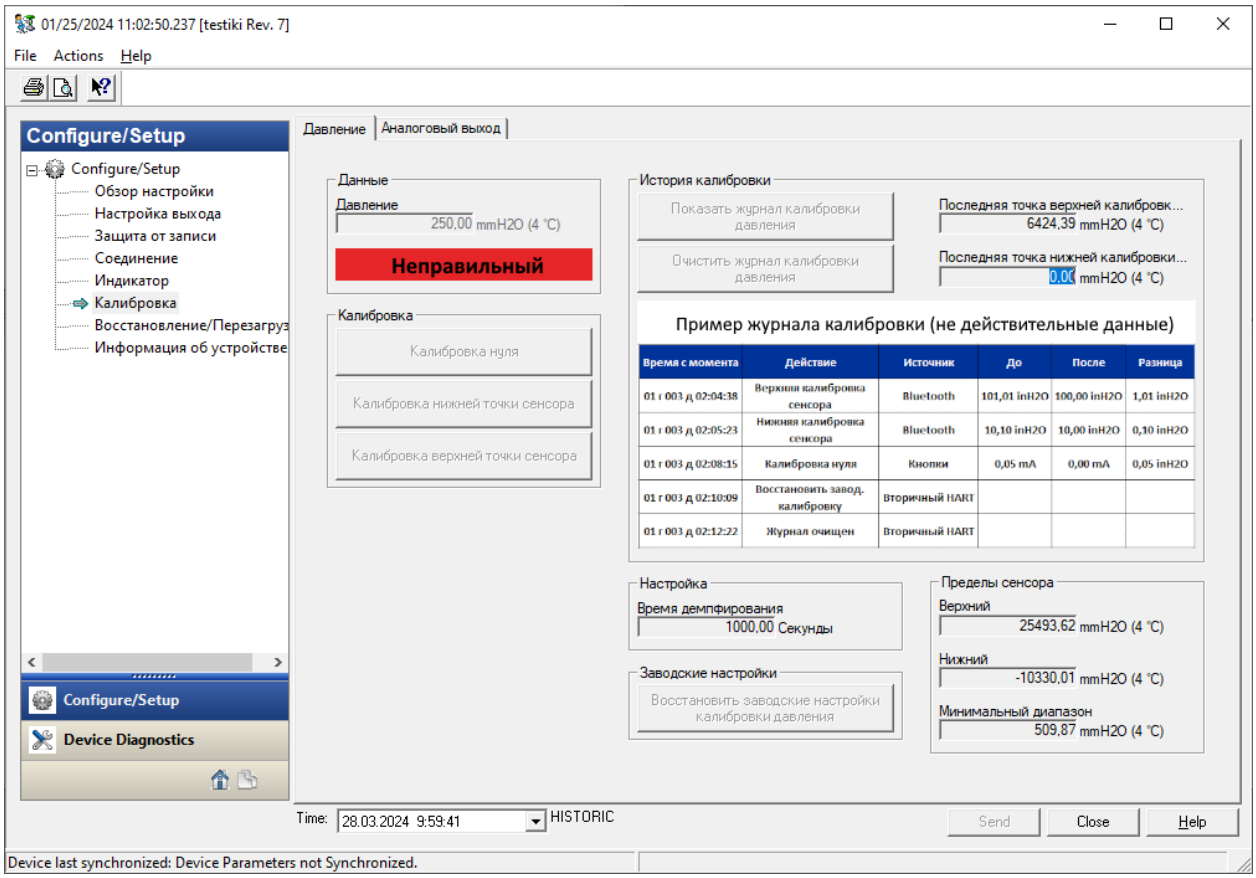

Рисунок 3.16 – Экранная форма калибровки давления

На рисунке 3.17 представлено окно с калибровкой аналогового выхода (токовой петли). К функциям относятся:

1. Проверка токовой петли.
2. Аналоговая калибровка.
3. Показать и очистить журнал калибровки аналогового выхода.
4. Восстановить заводские настройки аналоговой калибровки.

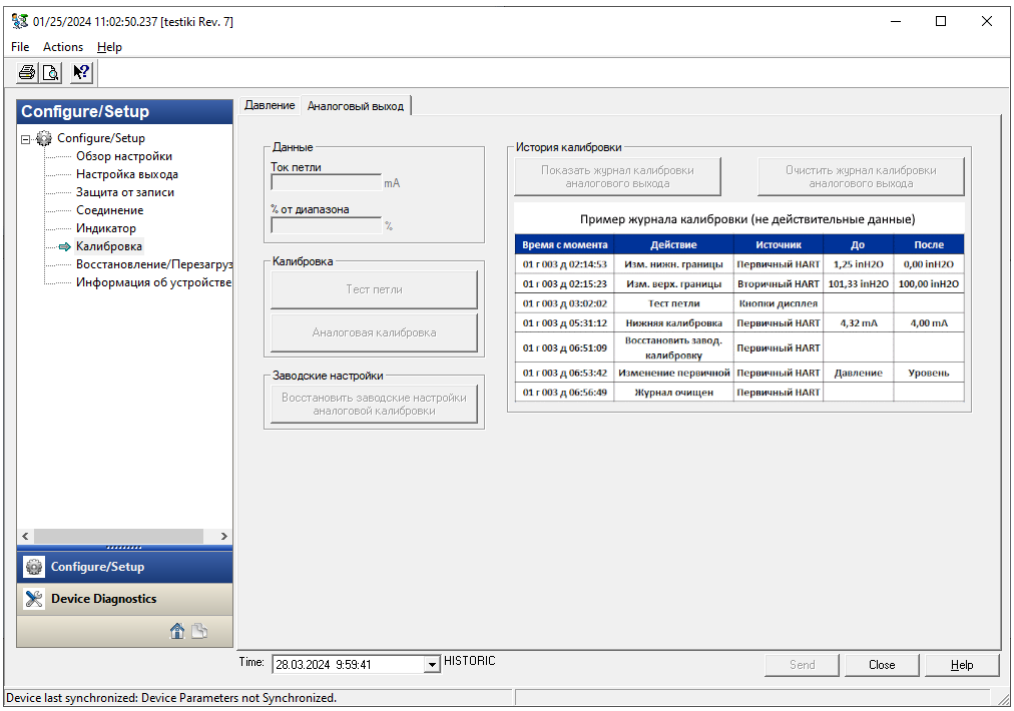

Рисунок 3.17 – Экранная форма калибровки давления

На рисунках 3.18-3.21 представлены окна с информацией об устройстве. На вкладке «Идентификация» (рисунок 3.18) отображается заводская информация о датчике.

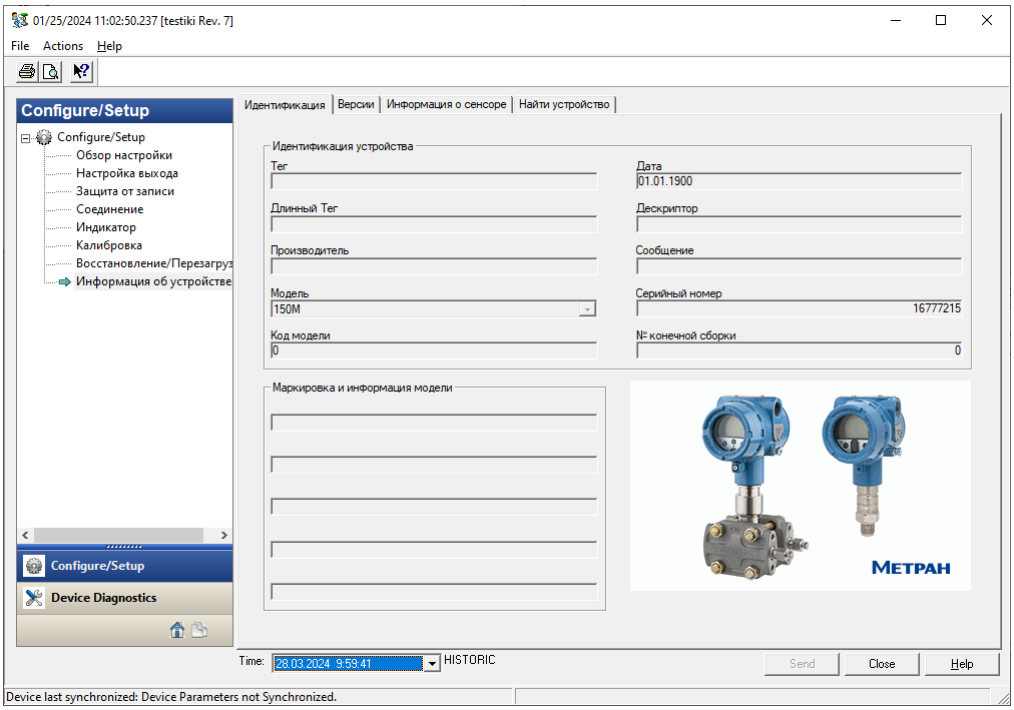

Рисунок 3.18 – Экранная форма информации об устройстве (идентификация)

На рисунке 3.19 представлена вкладка «Версии», на которой отражены номера версий используемого программного обеспечения и устройств.

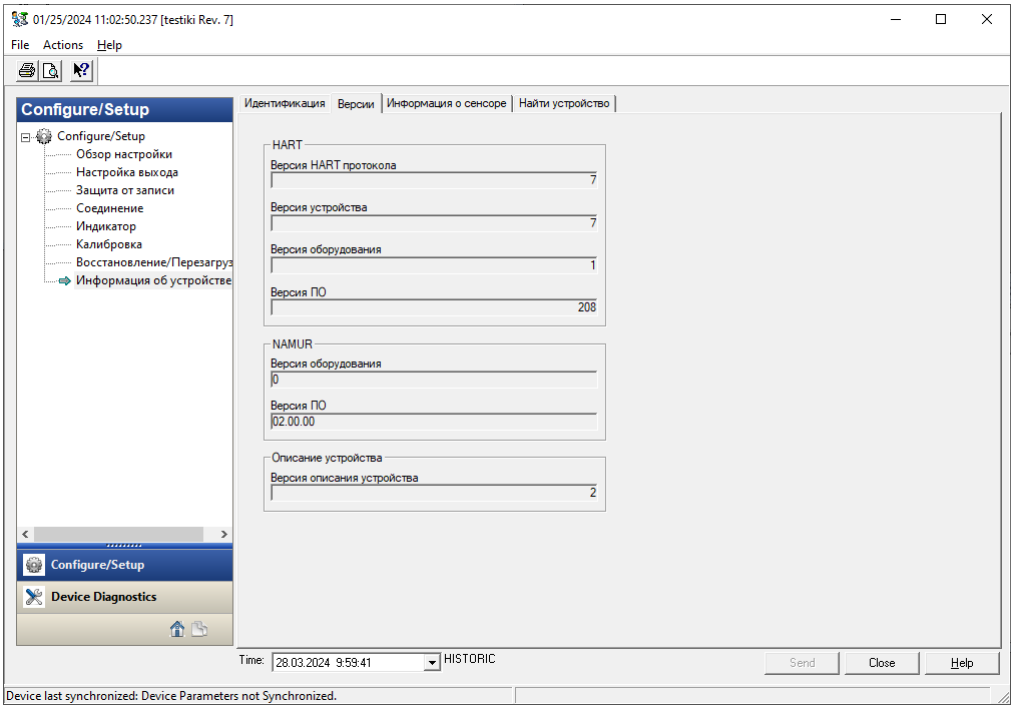

Рисунок 3.19 – Экранная форма информации об устройстве (версии)

На рисунке 3.20 представлена вкладка «Информация о сенсоре», на которой отражены сенсорные данные.

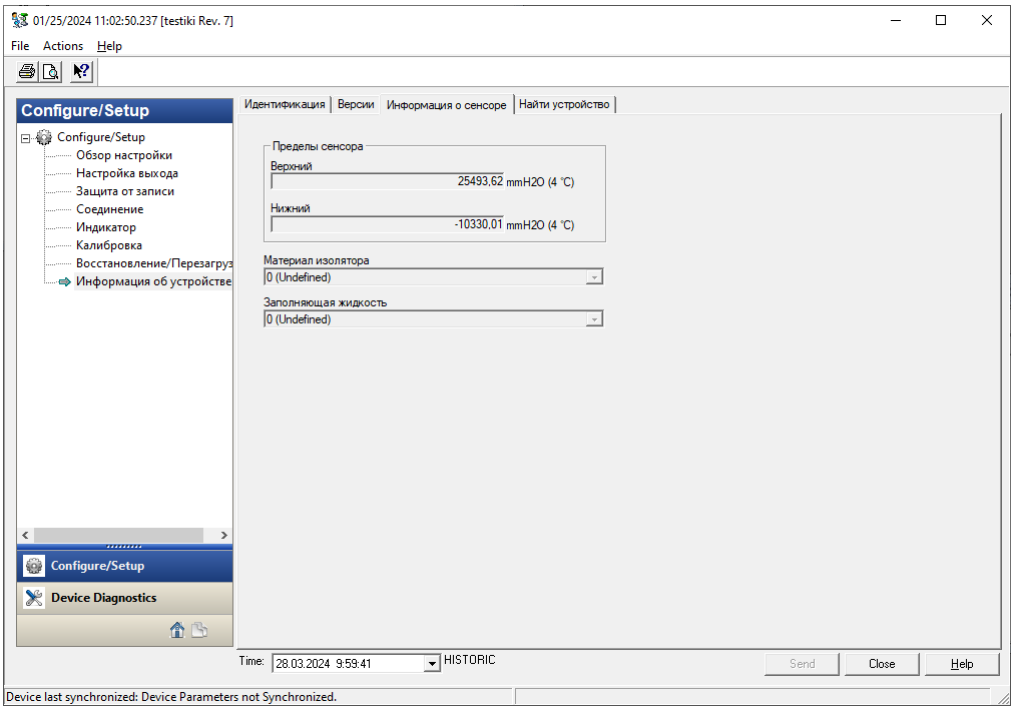

Рисунок 3.20 – Экранная форма информации об устройстве (информация о сенсоре)

На рисунке 3.21 представлена вкладка «Найти устройство», в ней есть возможность вызвать функцию по поиску прибора.

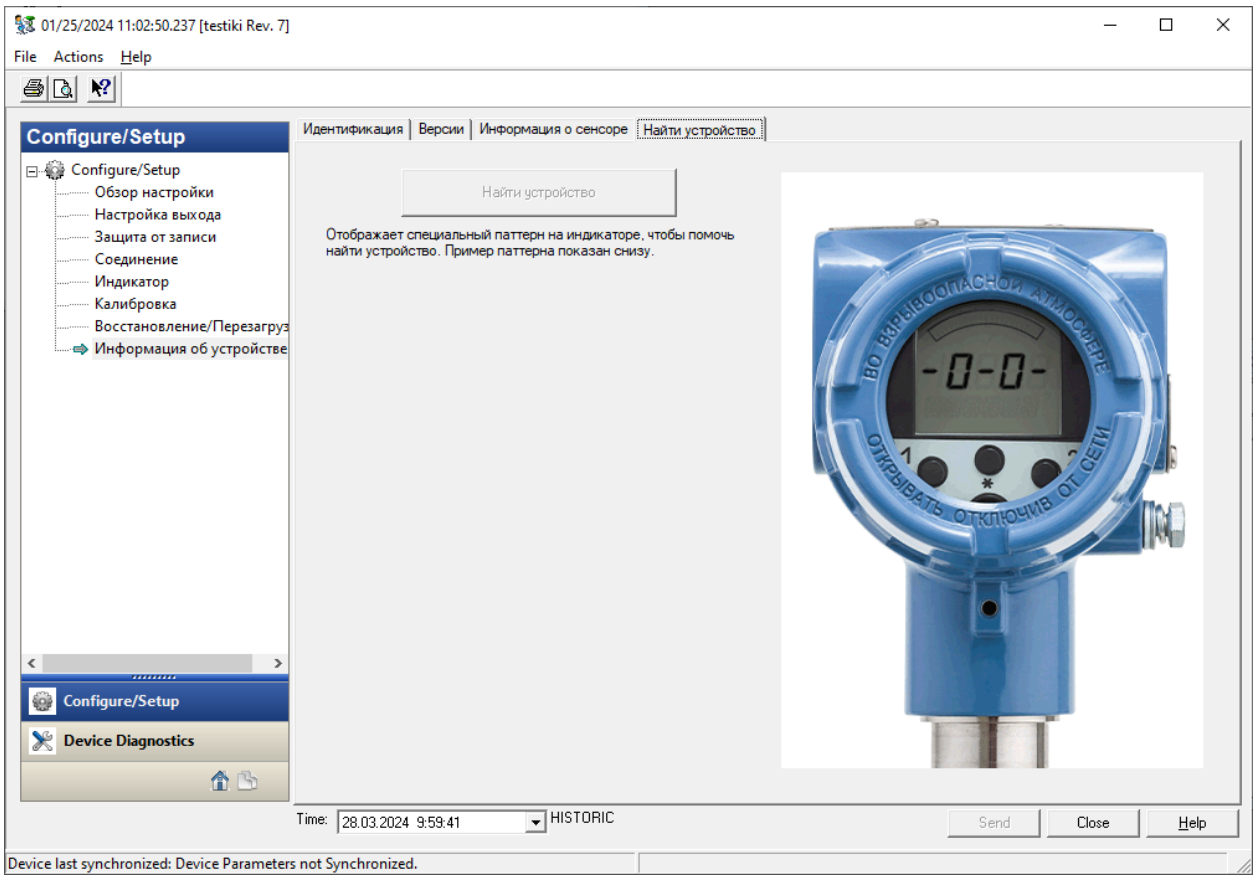

Рисунок 3.21 – Экранная форма информации об устройстве (найти устройство)

Исходный код представлен в приложении А и Б.

Вывод по разделу три

В рамках данного раздела разработан программный код драйвера для датчика давления. Реализованы все необходимые функции для настройки, диагностики и отслеживания технологических переменных.

4. ТЕСТИРОВАНИЕ И ДОРАБОТКА ПРОГРАММНОГО КОДА

В данном разделе необходимо провести тестирование всего функционала, который был разработан в рамках третьего раздела по написанию программного кода. Помимо этого, следует проверить корректность отображения всех окон пользовательского интерфейса: их непосредственное наличие, отображение всех функциональных элементов и их удобное для пользователя расположение.

Тестирование начнется с проверки отображения пользовательского меню и окон. Меню представляет из себя три основные группы: настройка, диагностика и технологические переменные (рисунки 4.1, 4.2, 4.3 соответственно), то есть настройка, диагностика и переменные. Список всех окон и результаты тестирования представлены в таблице 8.

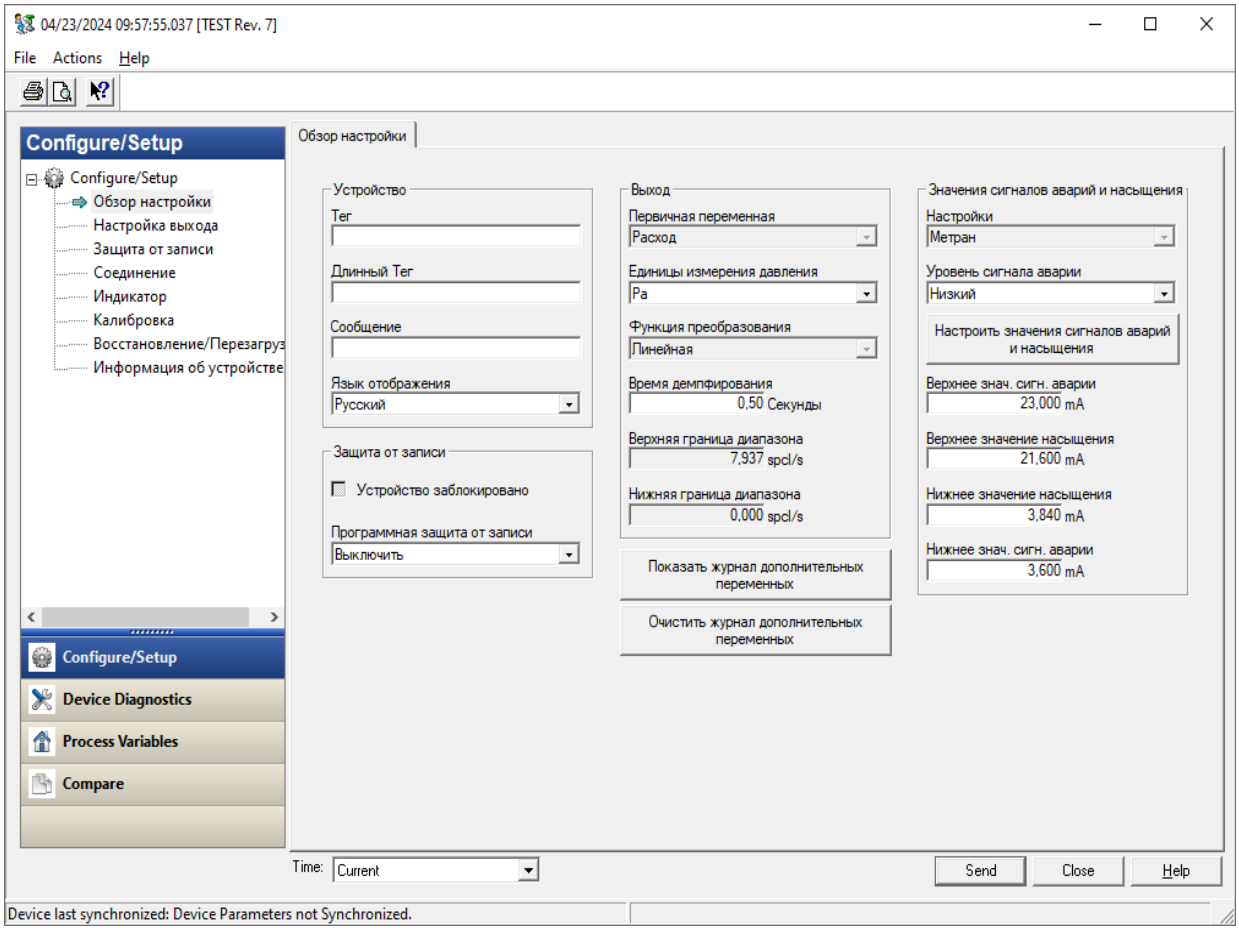

Рисунок 4.1 – Настройка

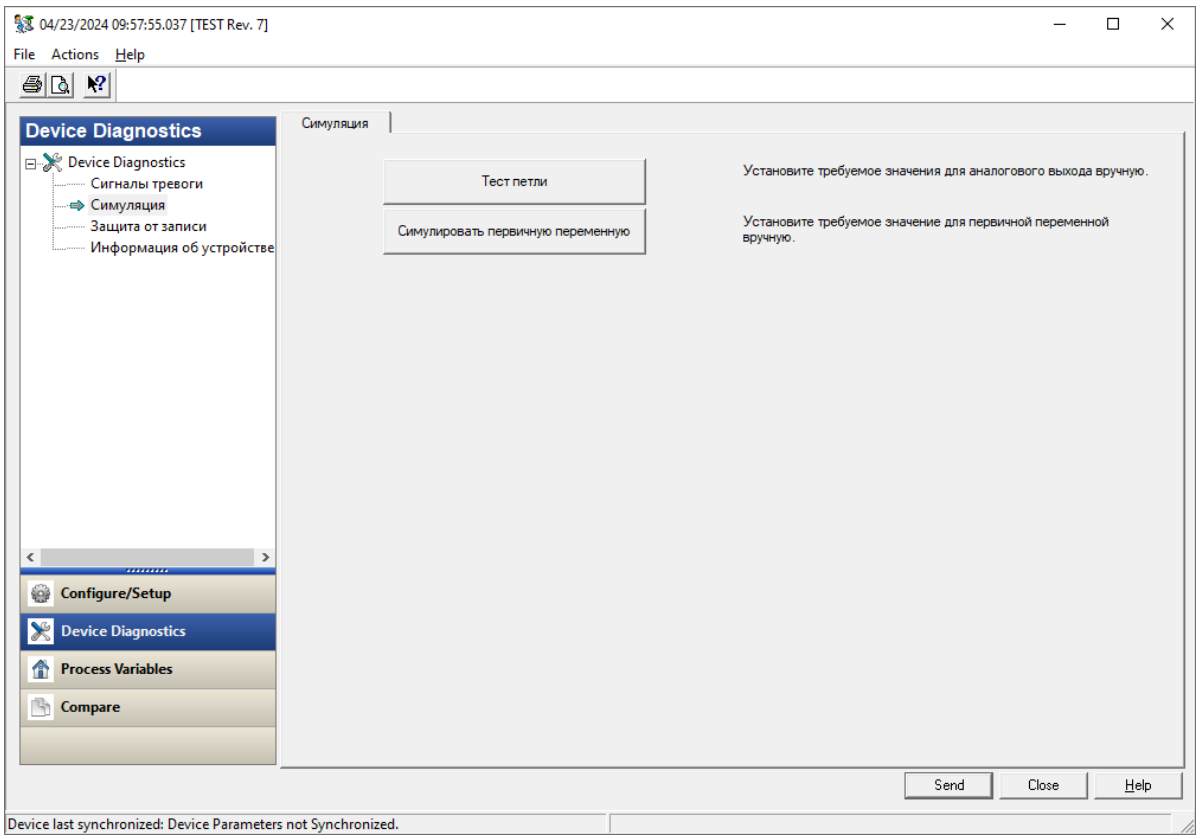

Рисунок 4.2 – Диагностика

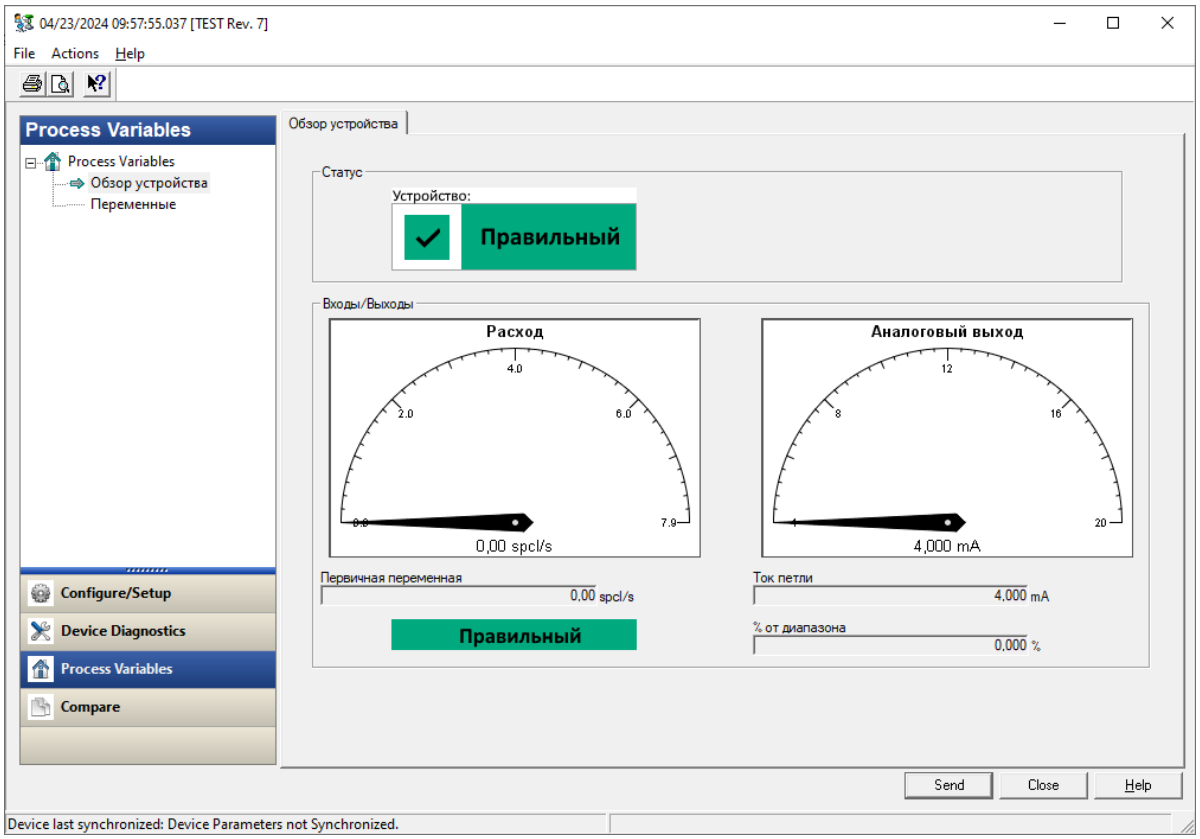

Рисунок 4.3 – Технологические переменные

Таблица 8 – Тестирование экранов

Наименование	Результат
Технологические переменные	
Обзор устройства	Успешно
Переменные	Успешно
Настройка	
Обзор настройки	Успешно
Настройки выхода – аналоговый выход	Успешно
Настройки выхода – давление	Успешно
Настройки выхода – расход	Успешно
Настройки выхода – счетчик расхода	Успешно
Настройки выхода – уровень	Успешно
Настройки выхода – объем	Успешно
Настройки выхода – температура сенсора	Успешно
Настройки выхода – температура платы	Успешно
Настройки выхода – порог предупреждения 1	Успешно
Настройки выхода – порог предупреждения 2	Успешно

Продолжение таблицы 8

Наименование	Результат
Настройки выхода – HART	Успешно
Защита от записи	Успешно
Соединение	Успешно
Индикатор	Успешно
Калибровка – давление	Успешно
Калибровка – аналоговый выход	Успешно
Восстановление/перезагрузка	Успешно
Информация об устройстве – идентификация	Успешно
Информация об устройстве – версии	Успешно
Информация об устройстве – информация о сенсоре	Успешно
Информация об устройстве – найти устройство	Успешно
Диагностика	
Сигналы тревоги – активные сигналы тревоги	Успешно
Сигналы тревоги – история	Успешно
Сигналы тревоги – порог предупреждения 1	Успешно
Сигналы тревоги – порог предупреждения 2	Успешно
Сигналы тревоги – диагностика целостности петли	Успешно
Симуляция	Успешно
Защита от записи	Успешно
Информация об устройстве – идентификация	Успешно
Информация об устройстве – версии	Успешно
Информация об устройстве – информация о сенсоре	Успешно

Информация об устройстве – найти устройство	Успешно

Проверка показала, что все пункты меню, а также пользовательские окна отображаются корректно.

Далее необходимо проверить работу всех функций драйвера. Их список и результаты тестирования представлены в таблице 9.

Таблица 9 – Тестирование функций

Наименование	**Результат**
Калибровка верхней точки сенсора	Успешно
Калибровка нижней точки сенсора	Успешно
Калибровка нуля	Успешно
Восстановить заводские настройки калибровки давления	Успешно
Настроить порог предупреждения 1	Успешно
Очистить журнал порога предупреждения 1	Успешно
Настроить порог предупреждения 2	Успешно
Очистить журнал порога предупреждения 2	Успешно
Сбросить счетчик расхода	Успешно
Настроить путем подачи давления	Успешно
Аналоговая калибровка	Успешно
Восстановить заводские настройки аналоговой калибровки	Успешно
Изменить адрес опроса	Успешно
Отрегулировать уровень	Успешно
Сбросить регулировку уровня	Успешно
Тест петли	Успешно
Настроить уровень	Успешно
Перезапуск устройства	Успешно
Сконфигурировать резервуар	Успешно
Показать журнал калибровки аналогового выхода	Успешно
Очистить журнал калибровки аналогового выхода	Успешно
Показать журнал калибровки давления	Успешно
Очистить журнал калибровки давления	Успешно
Показать последние 10 записей в журнале диагностики	Успешно
Показать последние 100 записей в журнале диагностики	Успешно
Очистить журнал диагностики	Успешно
Настроить значения сигналов аварий и насыщения	Успешно

Все функций, представленные в таблице 9, работают справно.

Отдельный этап тестирования – проверка уведомлений при сбоях и достижении установленных пользователем значений. Их типы можно разделить на следующие группы: сбой, проверка, обслуживание, контроль значений. Результаты проверки приведены в таблице 10.

Таблица 10 – Тестирование уведомлений

Наименование	**Результат**	**Комментарий**
Сбой		
Отказ электронной платы	Успешно	
Отказ сенсора	Успешно	
Несовместимый сенсор	Успешно	
Проверка		
Моделируемая первичная переменная (моделирование активно)	Успешно	
Статус моделирования активен	Успешно	
Токовая петля откалибрована	Успешно	
Контроль значений		
Давление выходит за пределы	Успешно	
Температура датчика/модуля выходит за пределы	Успешно	
Температура платы выходит за допустимые пределы	Успешно	
Предупреждение о смещении датчика давления	Успешно	
Обслуживание		
Оповещение процесса 1	Провал	При любых значениях выдает ошибку
Оповещение процесса 2	Провал	При любых значениях выдает ошибку
Кнопка застряла	Успешно	
Данные датчика не обновляются	Успешно	

Одно из предупреждений, установка порогового значения предупреждения, работает некорректно. Какие бы данные не были введены пользователем, возвращается ошибка, сообщение о которой представлено на

рисунке 4.4. Необходимо исправить недочет и проверить работу нового варианта реализации.

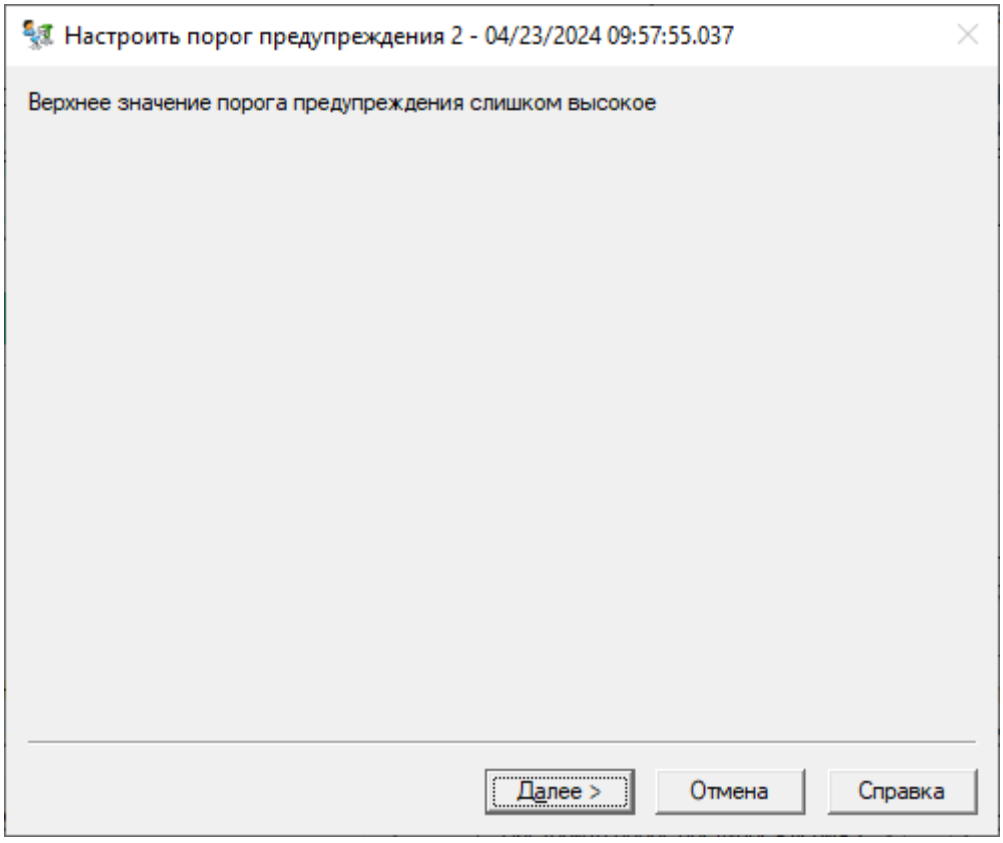

Рисунок 4.4 – Сообщение об ошибке

После поиска ошибки в коде, было обнаружено, что в одной из команд возвращалось неправильное значение при успешном ее выполнении (листинг…). Код 0, который сообщает об отсутствии сбоев и ошибок, возвращал параметр кода 9 (highAlertValueTooLarge_string). Необходимо установить «no_command_specific_errors_temp» для правильной работы.

Листинг 16 – Коды ответов

```
RESPONSE_CODES
{
    0,  SUCCESS,              highAlertValueTooLarge_string;
    2,  DATA_ENTRY_ERROR,     invalidSelectionProcessAlertEtAl_string;
    5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
    6,  MISC_ERROR,           deviceSpecificCommandErrorNVFailure_string;
    7,  MODE_ERROR,           in_write_protect_mode_temp;
    9,  DATA_ENTRY_ERROR,     highAlertValueTooLarge_string;
    10, DATA_ENTRY_ERROR,     highAlertValueTooSmall_string;
    11, DATA_ENTRY_ERROR,     lowAlertValueTooLarge_string;
    12, DATA_ENTRY_ERROR,     lowAlertValueTooSmall_string;
    13, DATA_ENTRY_ERROR,     timeDelayDeadbandTooLarge_string;
```

```
    15, DATA_ENTRY_ERROR,     timeDelayDeadbandTooSmall_string;
    16, MODE_ERROR,           accessRestricted_string;
}
```

После правок в коде проведены тесты уведомлений еще раз, чтобы убедиться в их исправности. В результате тестирования все работает исправно.

Вывод по разделу четыре

В рамках заключительного раздела проведено тестирование всего функционала драйвера. Тесты были разделены на три этапа: проверка отображения окон, функциональной работы и уведомлений.

Ошибка возникла во время третьего этапа – не работало уведомление о выходе за пределы значений при измерении. Недочеты были исправлены, а функция протестирована заново.

После тестов и исправлений драйвер можно считать пригодным к использованию, так как проверены все функции, а также удобство пользовательского взаимодействия с программой.

ЗАКЛЮЧЕНИЕ

В рамках выполнения выпускной квалифицированной работы разработан драйвер для датчика давления «Метран-150».

В работе рассмотрена задача по настройке, калибровке и прочему функционалу, связанному с обслуживанием и эксплуатацией датчиков давления. Разработанный драйвер предлагает решение данного вопроса.

По ходу работы были решены следующие задачи:

1. Проведен аналитический обзор научно-технической, нормативной и методической литературы по тематике работы.
2. Сформулированы требования, разработана архитектуры драйвера.
3. Разработан программный код.
4. Протестированы все функции драйвера.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Цифровизация промышленности. – https://clck.ru/3BA3Jh.
2. Тензометрический метод. – https://www.ngpedia.ru/id149312p2.html.
3. Коновалов Р.С. ВЫСОКОТЕМПЕРАТУРНЫЕ ДАТЧИКИ ДАВЛЕНИЯ / Р.С. Коновалов, А.А. Львов // Труды Международного симпозиума «Надежность и качество». – 2014. – №1.
4. Кремниевый тензорезисторный преобразователь давления. – https://sensorse.com/kremnievyi-preobrazovatel-davleniya.html.
5. Васильев С.И. Датчики систем автоматизации технологических процессов бурения нефтяных и газовых скважин : справочное пособие / С.И. Васильев, Л.А. Лапушова. – М.: Издательский дом Академии Естествознания, 2016. – 138 с.
6. Солодовников В.В. Устройства и элементы систем автоматического регулирования и управления. Техническая кибернетика. Книга 1. Измерительные устройства, преобразующие элементы и устройства / В.В. Солодовников – М.: Машиностроение, 1973. – 671 с.
7. Замалетдинова Э.Ю. СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ ИЗМЕРЕНИЯ ДАВЛЕНИЯ / Э.Ю. Замалетдинова, А.И. Егорычев // Вестник Казанского технологического университета. – 2014. – №8.
8. Михеев В.П. Датчики и детекторы : учебное пособие / В.П. Михеев, А.В. Просандеев – М.: МИФИ, 2007. – 172 с.
9. Финогенов Н.Ю. СПОСОБЫ ИЗМЕРЕНИЯ ДАВЛЕНИЯ ЖИДКОЙ СРЕДЫ ПРИ ПОМОЩИ РАЗЛИЧНЫХ ТИПОВ ДАТЧИКОВ ДАВЛЕНИЯ / Н.Ю. Финогенов // Вестник науки. – 2019. – Том 1. – №11.
10. Крутолапов А.С. Анализ концепции построения сетей информационного обмена / А.С. Крутолапов, Д.А. Сычев // Современные проблемы гражданской защиты. – 2012. – №4.
11. Лихтциндер Б.Я. СЕТИ ETHERNET С ДЕТЕРМИНИРОВАННЫМИ ЗАДЕРЖКАМИ / Б.Я. Лихтциндер // Вестник Самарского государственного технического университета. Серия:

Технические науки. – 2022. – Том 30. – №3.

12. Черепанов Н.В. Промышленный "Интернет вещей" на предприятии / Н.В. Черепанов // Инновации и инвестиции. – 2019. – №10.

13. Закалюжный А.А. Развитие беспроводных сетей как средство контроля и управления удаленными системами / А.А. Закалюжный // Молодой исследователь Дона. – 2018. – №4.

14. Current Signal Systems. – https://www.allaboutcircuits.com/textbook/direct-current/chpt-9/current-signal-systems/.

15. 2.4. Интерфейс "токовая петля". – https://www.reallab.ru/bookasutp/2-promishlennie-seti-i-interfeisi/2-4-interfeis-tokovaya-petlya/.

16. HART – DIGITAL TRANSFORMATION FOR ANALOG INSTRUMENTS. – https://www.fieldcommgroup.org/technologies/hart.

17. Денисенко В.В. HART-протокол: общие сведения и принципы построения сетей на его основе / В.В. Денисенко // Современные технологии автоматизации. – 2010. – №3. – С. 94-97.

18. What is HART Protocol ?. – https://instrumentationtools.com/what-is-hart-protocol/.

19. HART-протокол. – https://www.cta.ru/cms/f/343216.pdf.

20. HART Communication Protocol Specification. – https://library.fieldcommgroup.org/20013/TS20013/7.9/#page=2.

21. Чепурной А.А. CAN-технологии: для начинающих и принимающих решение / А.А. Чепурной // Компоненты и Технологии. – 2002. – №21.

22. Преображенский Н.Б. Контроллер периферии с Profibus dp интерфейсом / Н.Б. Преображенский, Ю.А. Холопов, Дам Чонг Нам // Международный научно-исследовательский журнал. – 2014. – №12-1.

23. Гусев С.В. Краткий экскурс в историю промышленных сетей / С.В. Гусев // Компоненты и Технологии. – 2001. – №12.

24. Гайнуллина А.А. Особенности организации передачи данных между программируемыми логическими контроллерами по протоколу Modbus / А.А. Гайнуллина, А.Д. Байтимиров // Вестник Казанского технологического университета. – 2013. – Том 16. – №23.

25. Промышленные сети (мировой рынок) Industrial Networks. – https://clck.ru/3BA3LQ.

26. ГОСТ Р МЭК 61511-1-2018. ГОСТ Р МЭК «Безопасность функциональная. Системы безопасности приборные для промышленных процессов. Часть 1. Термины, определения и технические требования». – М.: Стандартинформ, 2018. – 74 с.

ПРИЛОЖЕНИЯ

Приложение А

Листинг А.1 – Исходный код методов драйвера

```
/*****************************************************************************
*  FILENAME: 997207_methods.dd
*
*  DESCRIPTION: This file contains DD/FDI methods (including on navigation
screens and actions)
*****************************************************************************/

/*****************************************************************************
* Method: Range by Applying Pressure
* Description: Small Screen Method
*****************************************************************************/
METHOD rangeByApplyingPressure_method
{
    LABEL rangeByApplyingPressure_string;
    HELP rangeByApplyingPressure_help;
    DEFINITION
    {
        char status[STATUS_SIZE];
        int result;

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        CHECK_LOCK_SW_SECURITY_SWITCH;

        ACKNOWLEDGE(warningLoopRemoved_string);
        add_abort_method(returnToNormal_method);

        do
        {
            result = SELECT_FROM_LIST(selectTheRangeValue_string,
rangeValueToSet_string);
            switch(result)
            {
```

```
                case 0:
                    ACKNOWLEDGE(applyTheProcessValue4_string);
                    SEND_AND_PASS_ON_0_OR_14(37)
                    SEND_AND_FAIL_IF_ERROR(15)
                    ACKNOWLEDGE(setRangeValue_string);
                    break;
                case 1:
                    ACKNOWLEDGE(applyTheProcessValue20_string);
                    SEND_AND_FAIL_IF_ERROR(36)
                    SEND_AND_FAIL_IF_ERROR(15)
                    ACKNOWLEDGE(setRangeValue20_string);
                    break;
                default:
                    process_abort();
                    break;
            }
        }
        while(result != 2);

    }
}

/*********************************************************************
* Method: Configure Alarm and Saturation Values
* Description: Small Screen Method
*********************************************************************/
METHOD configureAlarmAndSaturationValues_method
{
    LABEL configureAlarmAndSaturationValues_string;
    HELP configureAlarmAndSaturationValues_help;
    DEFINITION
    {
        char status[STATUS_SIZE];
        int result;

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        CHECK_LOCK_SW_SECURITY_SWITCH;
```

```
        ACKNOWLEDGE(warningLoopRemoved_string);
        add_abort_method(returnToNormal_method);

        iassign(alarmSaturationSetting_local, alarmSaturationSetting);
        fassign(methodHighAlarm_local, methodHighAlarm);
        fassign(methodHighSaturation_local, methodHighSaturation);
        fassign(methodLowSaturation_local, methodLowSaturation);
        fassign(methodLowAlarm_local, methodLowAlarm);
        save_values();

        do
        {
            GET_DEV_VAR_VALUE(selectDesiredAlarmAndSaturation_string,
alarmSaturationSetting_local);
            if(alarmSaturationSetting_local == ALARM_SATURATION_CUSTOM)
            {
                GET_DEV_VAR_VALUE(enterTheHighAlarm_string,
methodHighAlarm_local);
                GET_DEV_VAR_VALUE(enterTheHighSaturation_string,
methodHighSaturation_local);
                GET_DEV_VAR_VALUE(enterTheLowSaturation_string,
methodLowSaturation_local);
                GET_DEV_VAR_VALUE(enterTheLowAlarm_string,
methodLowAlarm_local);
                result =
SELECT_FROM_LIST(setAlarmAndSaturationValues_string,
continueStartOver_string);
            }
            else
            {
                result = 0;
            }
        }
        while(result == 1);

        iassign(alarmSaturationSetting, alarmSaturationSetting_local);
        fassign(methodHighAlarm, methodHighAlarm_local);
        fassign(methodHighSaturation, methodHighSaturation_local);
        fassign(methodLowSaturation, methodLowSaturation_local);
        fassign(methodLowAlarm, methodLowAlarm_local);
```

```
        SEND_AND_FAIL_IF_ERROR(140)
        SEND_CMD_TRANS(142, 1)

        process_abort();
    }
}

/*****************************************************************************
* Method: Clear Totalizer
* Description: Small Screen Method
*****************************************************************************/
METHOD clearTotalizer_method
{
    LABEL clearTotalizer_string;
    HELP clearTotalizer_help;
    DEFINITION
    {
        char status[STATUS_SIZE];

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        CHECK_LOCK_SW_SECURITY_SWITCH;

        ACKNOWLEDGE(warningLoopRemoved_string);
        add_abort_method(returnToNormal_method);

        ACKNOWLEDGE(theCurrentTotalizerValue_string);

        tSetToZeroOption = CLEAR_TOTALIZER_VALUE;
        SEND_AND_FAIL_IF_ERROR(154)

        ACKNOWLEDGE(theTotalizerValueHasBeenSet_string);

        process_abort();
    }
}

/*****************************************************************************
```

```
* Method: Configure Tank
* Description: Large Screen Method
*****************************************************************************/
METHOD configureTank_method
{
    LABEL configureTank_string;
    HELP configureTank_help;
    DEFINITION
    {
        char status[STATUS_SIZE];
        long select;
        float maxLevel;
        float pointLevel;
        float pointVolume;
        float length;
        int point;
        unsigned int failedWriting;
        unsigned int errorPoint;
        select = 0;

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        CHECK_LOCK_SW_SECURITY_SWITCH;

        ACKNOWLEDGE(warningLoopRemoved_string);
        add_abort_method(returnToNormal_method);

        iassign(volumeSetupTankWriteType_local, volumeSetupTankReadType);
        //Local variables are used for units here because Simatic PDM v9.2
SP1 does not write user-selected units if it is a device variable.
        iassign(volumeTankUnits_local, volumeConfigUnits);
        iassign(levelTankUnits_local, levelConfigUnits);
        save_values();
        MenuDisplay(selectInformationForTank_dialog, next_string, select);
        iassign(volumeSetupTankWriteType, volumeSetupTankWriteType_local);

        iassign(device_variable_code, VOLUME_CODE);
        iassign(deviceVariables[VOLUME_CODE].CONFIG_UNITS,
```

```
volumeTankUnits_local);
        SEND_AND_FAIL_IF_ERROR(53)
        iassign(device_variable_code, LEVEL_CODE);
        iassign(deviceVariables[LEVEL_CODE].CONFIG_UNITS,
levelTankUnits_local);
        SEND_AND_FAIL_IF_ERROR(53)

        if(volumeSetupTankWriteType != TANK_TYPE_CUSTOM)
        {
            if(volumeSetupTankWriteType != TANK_TYPE_SPHERE)
            {
                MenuDisplay(noCustomConfigureTank_dialog, next_string,
select);
                PUT_MESSAGE(needAdditionalTime_string);
                if((context_collection.WRITE_TANK_LENGTH <= 0) ||
(context_collection.WRITE_TANK_RADIUS <= 0))
                {
                    ACKNOWLEDGE(valueShouldBeMoreThanZero_string);
                    process_abort();
                }
            }
            else
            {
                MenuDisplay(noCustomConfigureTankSphere_dialog,
next_string, select);
                PUT_MESSAGE(needAdditionalTime_string);
                if(context_collection.WRITE_TANK_RADIUS <= 0)
                {
                    ACKNOWLEDGE(valueShouldBeMoreThanZero_string);
                    process_abort();
                }
            }

            iassign(volumeSetupNumStrapWritePoints, 30);
            SEND_AND_FAIL_IF_ERROR(158)

            fassign(volumeSetupTankRadius_local,
volumeSetupTankWriteRadius);
            fassign(volumeSetupTankLength_local,
volumeSetupTankWriteLength);
            iassign(levelConfigUnitsCommand, levelConfigUnits);
```

```
            switch(levelConfigUnits)
            {
                case FEET:
                    iassign(volumeConfigUnitsCommand, CUBIC_FEET);
                    break;
                case METERS:
                    iassign(volumeConfigUnitsCommand, CUBIC_METERS);
                    break;
                case INCHES:
                    iassign(volumeConfigUnitsCommand, CUBIC_INCHES);
                    break;
                default:
                    //assign meters for easier units select of volume,
they will be sent in command 159 and values converted back by device
                    iassign(levelConfigUnitsCommand, METERS);
                    iassign(volumeConfigUnitsCommand, CUBIC_METERS);
                    //convert values to meters to match units (for
calculation)
                    CONVERT_TO_METERS(volumeSetupTankRadius_local)
                    CONVERT_TO_METERS(volumeSetupTankLength_local)
                    break;
            }

            switch(volumeSetupTankWriteType)
            {
                case TANK_TYPE_SPHERE:
                    SPHERE
                    break;
                case TANK_TYPE_HORIZONTAL_BULLET:
                    HORIZONTAL_BULLET
                    break;
                case TANK_TYPE_HORIZONTAL_CYLINDER:
                    HORIZONTAL_CYLINDER
                    break;
                case TANK_TYPE_VERTICAL_CYLINDER:
                    VERTICAL_CYLINDER
                    break;
                case TANK_TYPE_VERTICAL_BULLET:
                    VERTICAL_BULLET
                    break;
                default:
```

```
ACKNOWLEDGE(invalidSelectionConfigureVolumeSetup_string);
                    process_abort();
                    break;
            }

            writeLevelValueC7 = NaN;
            writeLevelValueD7 = NaN;
            writeVolumeValueC7 = NaN;
            writeVolumeValueD7 = NaN;
            for(writeStrappingPointSet_local = 0;
writeStrappingPointSet_local < 8; writeStrappingPointSet_local++)
            {
                SEND_AND_FAIL_IF_ERROR(159)
                //DELAY_TIME added to avoid RC10 on Instrument Inspector
                DELAY_TIME(1);
            }
        }
        else
        {
            iassign(volumeSetupNumStrapWritePoints_local,
volumeSetupNumStrapReadPoints);
            save_values();
            MenuDisplay(enterNumberOfStrappingPoints_dialog, next_string,
select);
            iassign(volumeSetupNumStrapWritePoints,
volumeSetupNumStrapWritePoints_local);
            fassign(volumeSetupTankWriteLength, 0);
            fassign(volumeSetupTankWriteRadius, 0);
            iassign(volumeConfigUnitsCommand, volumeUnits);
            iassign(levelConfigUnitsCommand, levelUnits);
            do
            {
                SEND_AND_FAIL_IF_ERROR(158)
                failedWriting = 0;
                errorPoint = 0;
                writeStrappingPointSet_local = 0;
                MenuDisplay(customConfigureTank_dialog, next_string,
select);
                PUT_MESSAGE(needAdditionalTime_string);
```

```
                for(point = 0; point < volumeSetupNumStrapWritePoints;
point++)
                {
                    writeLevel_array[point] =
writeLevelLocal_array[point];
                    writeVolume_array[point] =
writeVolumeLocal_array[point];
                    if((((point + 1) % 4) == 0) || ((point + 1) ==
volumeSetupNumStrapWritePoints))
                    {
                        send(159, status);
                        writeStrappingPointSet_local++;
                        if (status[STATUS_RESPONSE_CODE] != 0)
                        {
                            switch(status[STATUS_RESPONSE_CODE])
                            {
                                case 11:
                                    errorPoint = errorPoint + 1;

ACKNOWLEDGE(valueLevelIsNotMonotonic_string);
                                    break;
                                case 12:
                                    errorPoint = errorPoint + 2;

ACKNOWLEDGE(valueLevelIsNotMonotonic_string);
                                    break;
                                case 13:
                                    errorPoint = errorPoint + 3;

ACKNOWLEDGE(valueLevelIsNotMonotonic_string);
                                    break;
                                case 15:
                                    errorPoint = errorPoint + 4;

ACKNOWLEDGE(valueLevelIsNotMonotonic_string);
                                    break;
                                case 28:
                                    errorPoint = errorPoint + 1;

ACKNOWLEDGE(valueVolumeIsNotMonotonic_string);
```

```
                                    break;
                                case 29:
                                    errorPoint = errorPoint + 2;

ACKNOWLEDGE(valueVolumeIsNotMonotonic_string);
                                    break;
                                case 65:
                                    errorPoint = errorPoint + 3;

ACKNOWLEDGE(valueVolumeIsNotMonotonic_string);
                                    break;
                                case 66:
                                    errorPoint = errorPoint + 4;

ACKNOWLEDGE(valueVolumeIsNotMonotonic_string);
                                    break;
                                default:
                                    display_response_status(159,
status[STATUS_RESPONSE_CODE]);
                                    process_abort();
                                    break;
                            }
                            failedWriting = 1;
                            break;
                        }
                        errorPoint = errorPoint + 4;
                    }
                }
            }
            while(failedWriting == 1);
        }

        save_values();
        SEND_AND_FAIL_IF_ERROR(160)
        SEND_AND_FAIL_IF_ERROR(161)
        for(readStrappingPointSet_local = 0; readStrappingPointSet_local
<= 4; readStrappingPointSet_local++)
        {
            SEND_AND_FAIL_IF_ERROR(162)
        }
```

```
        process_abort();
    }
}

/*****************************************************************************
* Method: Simulate PV
* Description: Small Screen Method
*****************************************************************************/
METHOD simulatePV_method
{
    LABEL simulatePV_string;
    HELP simulatePV_help;
    DEFINITION
    {
        char status[STATUS_SIZE];

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        CHECK_LOCK_SW_SECURITY_SWITCH;

        ACKNOWLEDGE(warningLoopRemoved_string);
        add_abort_method(returnToNormal_method);

        SEND_AND_PASS_ON_0_OR_8(1)

        SEND_AND_FAIL_IF_ERROR(50)

        fassign(pvValue_local, PV.DEVICE_VARIABLE.DIGITAL_VALUE);
        save_values();
        GET_DEV_VAR_VALUE(enterFixedValueForPV_string, pvValue_local);

        iassign(device_variable_code, primary_variable_code);
        fassign(deviceVariables[device_variable_code].DIGITAL_VALUE,
pvValue_local);
        iassign(deviceVariables[device_variable_code].SIMULATED,
FIXED_STATUS);
        iassign(deviceVariables[device_variable_code].DIGITAL_UNITS,
PV.DEVICE_VARIABLE.DIGITAL_UNITS);
```

```
        iassign(deviceVariables[device_variable_code].DATA_QUALITY,
MANUAL_FIXED_STATUS);
        iassign(deviceVariables[device_variable_code].LIMIT_STATUS,
CONSTANT_STATUS);
        SEND_AND_FAIL_IF_ERROR(79)
        add_abort_method(returnPVToNormal_method);
        remove_abort_method(returnToNormal_method);

        SEND_AND_PASS_ON_0_OR_8(1)

        pvValue_local = PV.DEVICE_VARIABLE.DIGITAL_VALUE;
        ACKNOWLEDGE(pvIsFixed_string);

        process_abort();
    }
}

/*****************************************************************************
* Method: Loop Test
* Description: Small Screen Method
*****************************************************************************/
METHOD loopTest_method
{
    LABEL loopTest_string;
    HELP loopTest_help;
    DEFINITION
    {
        char status[STATUS_SIZE];
        int result;

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        CHECK_LOCK_SW_SECURITY_SWITCH;

        ACKNOWLEDGE(warningLoopRemoved_string);
        add_abort_method(returnAOToNormal_method);

        do
```

```
        {
            result = SELECT_FROM_LIST(chooseAnalogOutputLevel_string,
analogOutputLevelToSet_string);
            switch(result)
            {
                case 0:
                    fassign(aoValue_local, 4.000);
                    break;
                case 1:
                    fassign(aoValue_local, 20.000);
                    break;
                case 2:
                    SEND_CMD_TRANS(142, 0)
                    SEND_AND_FAIL_IF_ERROR(15)
                    if(PV.DAQ.ALARM_CODE == 0)
                    {
                        fassign(aoValue_local, methodHighAlarm);
                    }
                    else
                    {
                        fassign(aoValue_local, methodLowAlarm);
                    }
                    break;
                case 3:
                    fassign(aoValue_local, loopCurrent);
                    save_values();
                    GET_DEV_VAR_VALUE(fixAnalogOutput_string,
aoValue_local);
                    //Added condition to show error message when user
tries to write 0 in custom case
                    if(abs(aoValue_local) < EPSILON_VALUE)
                    {
                        ACKNOWLEDGE(Parameter_too_small_temp);
                        process_abort();
                    }
                    break;
                default:
                    process_abort();
                    break;
            }
```

```
            fassign(PV.DAQ.ANALOG_VALUE, aoValue_local);
            SEND_AND_FAIL_IF_ERROR(40)
            SEND_AND_FAIL_IF_ERROR(2)
            ACKNOWLEDGE(analogOutputIsFixed_string);
        }
        while(result != 4);
    }
}

/*****************************************************************************
* Method: Configure Process Alert 1
* Description: Small Screen Method
*****************************************************************************/
METHOD configureProcessAlert1_method
{
    LABEL configureProcessAlert1_string;
    HELP configureProcessAlert1_help;
    DEFINITION
    {
        CONFIGURE_PROCESS_ALERT(PROCESS_ALERT_1_ID)
    }
}

/*****************************************************************************
* Method: Configure Process Alert 2
* Description: Small Screen Method
*****************************************************************************/
METHOD configureProcessAlert2_method
{
    LABEL configureProcessAlert2_string;
    HELP configureProcessAlert2_help;
    DEFINITION
    {
        CONFIGURE_PROCESS_ALERT(PROCESS_ALERT_2_ID)
    }
}

/*****************************************************************************
* Method: Clear Process Alert 1 Log
* Description: Small Screen Method
*****************************************************************************/
```

```
METHOD clearProcessAlert1Log_method
{
    LABEL clearProcessAlert1Log_string;
    HELP clearProcessAlert1Log_help;
    DEFINITION
    {
        CLEAR_ALERT_LOG(PROCESS_ALERT_1_ID)
    }
}

/*****************************************************************************
* Method: Clear Process Alert 2 Log
* Description: Small Screen Method
*****************************************************************************/
METHOD clearProcessAlert2Log_method
{
    LABEL clearProcessAlert2Log_string;
    HELP clearProcessAlert2Log_help;
    DEFINITION
    {
        CLEAR_ALERT_LOG(PROCESS_ALERT_2_ID)
    }
}

/*****************************************************************************
* Method: Clear Pressure Log
* Description: Small Screen Method
*****************************************************************************/
METHOD clearPressureLog_method
{
    LABEL clearPressureLog_string;
    HELP clearPressureLog_help;
    DEFINITION
    {
        CLEAR_LOG(MIN_MAX_PRESSURE)
    }
}

/*****************************************************************************
* Method: Clear Sensor Temperature Log
* Description: Small Screen Method
```

```
*****************************************************************************/
METHOD clearSensorTemperatureLog_method
{
    LABEL clearSensorTemperatureLog_string;
    HELP clearSensorTemperatureLog_help;
    DEFINITION
    {
        CLEAR_LOG(MIN_MAX_SENSOR_TEMPERATURE)
    }
}

/*****************************************************************************
* Method: Clear Board Temperature Log
* Description: Small Screen Method
*****************************************************************************/
METHOD clearBoardTemperatureLog_method
{
    LABEL clearBoardTemperatureLog_string;
    HELP clearBoardTemperatureLog_help;
    DEFINITION
    {
        CLEAR_LOG(MIN_MAX_BOARD_TEMPERATURE)
    }
}

/*****************************************************************************
* Method: Adjust Level Reading
* Description: Small Screen Method
*****************************************************************************/
METHOD adjustLevelReading_method
{
    LABEL adjustLevelReading_string;
    HELP adjustLevelReading_help;
    DEFINITION
    {
        char status[STATUS_SIZE];

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();
```

```
        CHECK_LOCK_SW_SECURITY_SWITCH;

        ACKNOWLEDGE(warningLoopRemoved_string);
        add_abort_method(returnToNormal_method);

        fassign(actualLevel_local, levelValue);
        save_values();

        do
        {
            GET_DEV_VAR_VALUE(enterTheActualLevel_string,
actualLevel_local);
            fassign(actualLevel, actualLevel_local);
            SEND_CMD(156)
        }
        while((status[STATUS_RESPONSE_CODE] == 9) ||
(status[STATUS_RESPONSE_CODE] == 10));

        if(status[STATUS_RESPONSE_CODE] == 0)
        {
            SEND_AND_FAIL_IF_ERROR(157)
        }

        process_abort();
    }
}

/*****************************************************************************
* Method: Clear Level Adjustment
* Description: Small Screen Method
*****************************************************************************/
METHOD clearLevelAdjustment_method
{
    LABEL clearLevelAdjustment_string;
    HELP clearLevelAdjustment_help;
    DEFINITION
    {
        char status[STATUS_SIZE];

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
```

```
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        CHECK_LOCK_SW_SECURITY_SWITCH;

        ACKNOWLEDGE(warningLoopRemoved_string);
        add_abort_method(returnToNormal_method);

        ACKNOWLEDGE(currentLevelAdjustment_string);
        iassign(tSetToZeroOption, CLEAR_LEVEL_ADJUSTMENT);
        SEND_AND_FAIL_IF_ERROR(154)
        SEND_AND_FAIL_IF_ERROR(157)
        ACKNOWLEDGE(adjustmentHasBeenSetToZero_string);

        process_abort();
    }
}

/*****************************************************************************
* Method: Level Configurator
* Description: Large Screen Method
*****************************************************************************/
METHOD levelConfigurator_method
{
    LABEL levelConfigurator_string;
    HELP levelConfigurator_help;
    DEFINITION
    {
        char status[STATUS_SIZE];
        long select;
        select = 0;

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        CHECK_LOCK_SW_SECURITY_SWITCH;

        ACKNOWLEDGE(warningLoopRemoved_string);
        add_abort_method(returnToNormal_method);
```

```
        GET_DEV_VAR_VALUE(selectUnitInformation_string, levelConfigUnits);
        PUT_MESSAGE(needAdditionalTime_string);
        iassign(device_variable_code, LEVEL_CODE);
        SEND_AND_FAIL_IF_ERROR(53)
        SEND_CMD_TRANS(157, 0)

        GET_DEV_VAR_VALUE(selectTankConfiguration_string,
tankConfigurationList_local);
        if(tankConfigurationList_local == VENTED_TANK)
        {
            GET_DEV_VAR_VALUE(selectTechnology_string,
openTechnologyList_local);
            save_values();
            MenuDisplay(ventedTankLevelConfigurator_dialog, next_string,
select);
            CONVERT_TO_INCHES(methodLevelSetupMinLevel);
            CONVERT_TO_INCHES(methodLevelSetupMaxLevel);
            switch(openTechnologyList_local)
            {
                case DIRECT_MOUNT_TRANSMITTER:
                    methodLevelSetupPressureAtMinLevel =
methodLevelSetupMinLevel * processSpecificGravity_local;
                    methodLevelSetupPressureAtMaxLevel =
methodLevelSetupMaxLevel * processSpecificGravity_local;
                    break;
                case TRANSMITTER_WITH_CAPILLARY:
                    if(fillFluid_local == 24)
                    {

GET_DEV_VAR_VALUE(enterSpecificGravityOfCustomFillFluid_string,
specificGravity_local);
                    }
                    else
                    {
                        SPECIFIC_GRAVITY(fillFluid_local)
                    }
                    CONVERT_TO_INCHES(height_local)
                    methodLevelSetupPressureAtMinLevel =
(methodLevelSetupMinLevel * processSpecificGravity_local) - (height_local
* specificGravity_local);
                    methodLevelSetupPressureAtMaxLevel =
```

```
(methodLevelSetupMaxLevel * processSpecificGravity_local) - (height_local
* specificGravity_local);
                    break;
                default:
                    CONVERT_TO_INCHES(height_local)
                    methodLevelSetupPressureAtMinLevel =
(methodLevelSetupMinLevel * processSpecificGravity_local) - (height_local
* processSpecificGravity_local);
                    methodLevelSetupPressureAtMaxLevel =
(methodLevelSetupMaxLevel * processSpecificGravity_local) - (height_local
* processSpecificGravity_local);
                    break;
            }
        }
        else
        {
            GET_DEV_VAR_VALUE(selectTechnology_string,
closedTechnologyList_local);
            save_values();
            MenuDisplay(pressurizedTankLevelConfigurator_dialog,
next_string, select);
            CONVERT_TO_INCHES(methodLevelSetupMinLevel);
            CONVERT_TO_INCHES(methodLevelSetupMaxLevel);
            CONVERT_TO_INCHES(height_local);
            switch(closedTechnologyList_local)
            {
                case WET_LEG_IMPULSE_PIPING:
                    CONVERT_TO_INCHES(heightImpulse_local);
                    methodLevelSetupPressureAtMinLevel =
((methodLevelSetupMinLevel - height_local) * processSpecificGravity_local)
- (heightImpulse_local * legSpecificGravity_local);
                    methodLevelSetupPressureAtMaxLevel =
((methodLevelSetupMaxLevel - height_local) * processSpecificGravity_local)
- (heightImpulse_local * legSpecificGravity_local);
                    break;
                case CAPILLARY_SYSTEM:
                    if(fillFluid_local == 24)
                    {

GET_DEV_VAR_VALUE(enterSpecificGravityOfCustomFillFluid_string,
specificGravity_local);
```

```
                    }
                    else
                    {
                        SPECIFIC_GRAVITY(fillFluid_local)
                    }
                    methodLevelSetupPressureAtMinLevel =
(methodLevelSetupMinLevel * processSpecificGravity_local) - (height_local
* specificGravity_local);
                    methodLevelSetupPressureAtMaxLevel =
(methodLevelSetupMaxLevel * processSpecificGravity_local) - (height_local
* specificGravity_local);
                    break;
                default:
                    methodLevelSetupPressureAtMinLevel =
(methodLevelSetupMinLevel * processSpecificGravity_local) - (height_local
* processSpecificGravity_local);
                    methodLevelSetupPressureAtMaxLevel =
(methodLevelSetupMaxLevel * processSpecificGravity_local) - (height_local
* processSpecificGravity_local);
                    break;
            }
        }

        iassign(writeLevelUnits, INCHES);
        iassign(writePressureUnits, 1);
        SEND_CMD_TRANS(155, 0)
        SEND_CMD_TRANS(157, 1)

        ACKNOWLEDGE(levelConfigurationComplete_string);
        process_abort();
    }
}

/*****************************************************************************
* Method: Change Polling Address
* Description: Small Screen Method
*****************************************************************************/
METHOD changePollingAddress_method
{
    LABEL changePollingAddress_string;
    HELP changePollingAddress_help;
```

```
    DEFINITION
    {
        char status[STATUS_SIZE];

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        CHECK_LOCK_SW_SECURITY_SWITCH;

        ACKNOWLEDGE(warningLoopRemoved_string);
        add_abort_method(returnToNormal_method);

        iassign(pollingAddress_local, polling_address);
        save_values();

        GET_DEV_VAR_VALUE(enterTheDesiredPollingAddress_string,
pollingAddress_local);

        if(pollingAddress_local > MAX_POLLING_ADDRESS)
        {
            ACKNOWLEDGE(invalidSelectionPollingAddress_string);
        }
        else
        {
            iassign(polling_address, pollingAddress_local);
            if(polling_address == 0)
            {
                iassign(loop_current_mode, LOOP_CURRENT_MODE_ENABLE);
            }
            else
            {
                iassign(loop_current_mode, LOOP_CURRENT_MODE_DISABLE);
            }
            SEND_AND_FAIL_IF_ERROR(6)
        }

        process_abort();
    }
}
```

```
/*****************************************************************************
* Method: Device Restart
* Description: Small Screen Method
*****************************************************************************/
METHOD deviceRestart_method
{
    LABEL deviceRestart_string;
    HELP deviceRestart_help;
    DEFINITION
    {
        char status[STATUS_SIZE];

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        ACKNOWLEDGE(warningLoopRemoved_string);
        add_abort_method(returnToNormal_method);

        ACKNOWLEDGE(continuingRestartDevice_string);
        SEND_AND_FAIL_IF_ERROR(42)
        RETRY_ON_COMM_ERROR();
        DELAY(6, deviceBeingRestarted_string);
        process_abort();
    }
}

/*****************************************************************************
* Method: Analog Calibration
* Description: Small Screen Method
*****************************************************************************/
METHOD analogCalibration_method
{
    LABEL analogCalibration_string;
    HELP analogCalibration_help;
    DEFINITION
    {
        char status[STATUS_SIZE];
        int result;
        int isScaled;
```

```
        float scalingFactor;

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        CHECK_LOCK_SW_SECURITY_SWITCH;

        ACKNOWLEDGE(warningLoopRemoved_string);
        add_abort_method(returnToNormal_method);

        isScaled = SELECT_FROM_LIST(selectAnalogCalibration_string,
selectAnalogCalibrationVariants_string);
        if(isScaled == 0) // user selected "Digital to Analog Trim"
        {
            ACKNOWLEDGE(connectReferenceMeterAnalogCalibration_string);

            add_abort_method(returnAOToNormal_method);
            remove_abort_method(returnToNormal_method);

            AO_TRIM(TRIM_CMD_LOW, MIN_AO_VALUE)

            ACKNOWLEDGE(continueToSetAnalogOutputHigh_string);
            AO_TRIM(TRIM_CMD_HIGH, MAX_AO_VALUE)
        }
        else // user selected "Scaled Digital to Analog Trim"
        {
            fassign(scaledLow_local, MIN_AO_VALUE);
            fassign(scaledHigh_local, MAX_AO_VALUE);
            save_values();

            do
            {
                GET_DEV_VAR_VALUE(enterScaledOutputValueLow_string,
scaledLow_local);
                GET_DEV_VAR_VALUE(enterScaledOutputValueHigh_string,
scaledHigh_local);

                result = SELECT_FROM_LIST(trimWillBeScaled_string,
continueOrChangeScalePoints_string);
```

```
            }
            while(result != 0);

            scalingFactor = (20 - 4) / (scaledHigh_local -
scaledLow_local);

            ACKNOWLEDGE(continueToSetAnalogOutputLow_string);

            add_abort_method(returnAOToNormal_method);
            remove_abort_method(returnToNormal_method);

            AO_TRIM(TRIM_CMD_LOW, scaledLow_local)

            ACKNOWLEDGE(continueToSetAnalogOutputHigh_string);
            AO_TRIM(TRIM_CMD_HIGH, scaledHigh_local)
        }
        process_abort();
    }
}

/*****************************************************************************
* Method: Restore Analog Calibration
* Description: Small Screen Method
*****************************************************************************/
METHOD restoreAnalogCalibration_method
{
    LABEL restoreAnalogCalibration_string;
    HELP restoreAnalogCalibration_help;
    DEFINITION
    {
        char status[STATUS_SIZE];

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        CHECK_LOCK_SW_SECURITY_SWITCH;

        ACKNOWLEDGE(warningLoopRemoved_string);
        add_abort_method(returnToNormal_method);
```

```
        SEND_AND_FAIL_IF_ERROR(141)

        ACKNOWLEDGE(analogCalibrationRestored_string);
        process_abort();
    }
}

/*****************************************************************************
* Method: Clear Analog Calibration
* Description: Small Screen Method
*****************************************************************************/
METHOD clearAnalogCalibrationLog_method
{
    LABEL clearAnalogCalibrationLog_string;
    HELP clearAnalogCalibrationLog_help;
    DEFINITION
    {
        CLEAR_LOG(ANALOG_CALIBRATION_LOG)
    }
}

/*****************************************************************************
* Method: View Analog Calibration Log
* Description: Large Screen Method
*****************************************************************************/
METHOD viewAnalogCalibrationLog_method
{
    LABEL viewAnalogCalibrationLog_string;
    HELP viewAnalogCalibrationLog_help;
    DEFINITION
    {
        char status[STATUS_SIZE];
        int page;
        int letter;
        int indexArray;
        long select;
        select = 0;

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
```

```
        XMTR_IGNORE_ALL_RESPONSE_CODES();


        for(page = 0; page < 4; page++)
        {
            iassign(aoLogEntrySet, page);
            SEND_AND_FAIL_IF_ERROR(177)
            for(letter = 0; letter < 5; letter++)
            {
                indexArray = letter + (page * 5);
                analogOutputLogFound_array[indexArray] = blank_string;
                analogOutputLogLeft_array[indexArray] = blank_string;
                iassign(analogOutputLogAction_array[indexArray],
actionValueAO_array[letter]);
                iassign(analogOutputLogSource_array[indexArray],
sourceValueAO_array[letter]);

                if(actionValueAO_array[letter] != NO_LOGGED_ACTION_ID)
                {
                    GET_STRING_BY_SECONDS(timeSinceValueAO_array[letter]);
                    analogOutputLogTimeSince_array[indexArray] =
formattedDateTimeValue;
                    if((actionValueAO_array[letter] != LOG_CLEARED_ID) &&
(actionValueAO_array[letter] != RESTORE_FACTORY_TRIM_ID) &&
(actionValueAO_array[letter] != LOOP_TEST_ID)  &&
(actionValueAO_array[letter] != CHANGE_ALARM_SATURATION_ID) &&
(actionValueAO_array[letter] != CHANGE_ALARM_DIRECTION_ID)  &&
(actionValueAO_array[letter] != CHANGE_TRANSFER_FUNCTION_ID))
                    {
                        //defining the value only of the variable
foundDevVarValue_local, because precision for foundAOLocal_string and for
leftAOLocal_string define of the foundDevVarValue_local, also the value of
this variable is used when actionValueAO_array[letter] = PV_CHANGE_ID
                        if((actionValueAO_array[letter] ==
DIGITAL_TO_ANALOG_UPPER_TRIM_ID) || (actionValueAO_array[letter] ==
DIGITAL_TO_ANALOG_LOWER_TRIM_ID))
                        {
                            foundDevVarValue_local = LOOP_CURRENT_CODE;
                        }
                        else
                        {
```

```
                            foundDevVarValue_local =
foundDevVarValueAO_array[letter];
                        }

                        if(actionValueAO_array[letter] != PV_CHANGE_ID)
                        {
                            unitsCodeValue_local =
unitsCodeValueAO_array[letter];
                            foundAO_local = foundValueAO_array[letter];
                            leftAO_local = leftValueAO_array[letter];
                            analogOutputLogFound_array[indexArray] =
BUILD_MESSAGE(foundAOLocal_string);
                            analogOutputLogLeft_array[indexArray] =
BUILD_MESSAGE(leftAOLocal_string);
                        }
                        else
                        {
                            analogOutputLogFound_array[indexArray] =
BUILD_MESSAGE(foundDevVarValueLocal_string);
                            leftDevVarValue_local =
leftDevVarValueAO_array[letter];
                            analogOutputLogLeft_array[indexArray] =
BUILD_MESSAGE(leftDevVarValueLocal_string);
                        }
                    }
                }
                else
                {
                    analogOutputLogTimeSince_array[indexArray] =
blank_string;
                }
            }
        }

        save_values();
        MenuDisplay(analogOutputCalibrationLog_dialog, next_string,
select);

        process_abort();
    }
}
```

```
/*****************************************************************************
* Method: Configure Password
* Description: Small Screen Method
*****************************************************************************/
METHOD configurePassword_method
{
    LABEL configurePassword_string;
    HELP configurePassword_help;
    DEFINITION
    {
        char status[STATUS_SIZE];

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        SEND_AND_FAIL_IF_ERROR(196)

        CHECK_LOCK_SW_SECURITY_SWITCH
        GET_DEV_VAR_VALUE(ifProtectionEnabled_string,
loiPasswordProtection);
        if(loiPasswordProtection == PASSWORD_PROTECTION_ENABLED)
        {
            userLOIPassword_local = blank_string;
            save_values();
            GET_DEV_VAR_VALUE(enterNumericalPassword_string,
userLOIPassword_local);
            loiPassword = userLOIPassword_local;
        }
        else
        {
            loiPassword = defaultLOIPassword_string;
        }
        SEND_AND_FAIL_IF_ERROR(190)
        process_abort();
    }
}

/*****************************************************************************
* Method: Zero Sensor Trim
```

```
* Description: Small Screen Method
*****************************************************************************/
METHOD zeroSensorTrim_method
{
    LABEL zeroSensorTrim_string;
    HELP zeroSensorTrim_help;
    DEFINITION
    {
        char status[STATUS_SIZE];

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        CHECK_LOCK_SW_SECURITY_SWITCH;

        ACKNOWLEDGE(warningLoopRemoved_string);
        add_abort_method(returnToNormal_method);

        do
        {
            ACKNOWLEDGE(applyZeroInput_string);

            iassign(device_variable_code, PRESSURE_CODE);
            SEND_CMD(52)

        }
        while(status[STATUS_RESPONSE_CODE] != 0);

        iassign(device_variable_trim_code, PRESSURE_CODE);
        SEND_AND_FAIL_IF_ERROR(80)

        ACKNOWLEDGE(zeroTrimSucceeded_string);

        process_abort();
    }
}

/*****************************************************************************
* Method: Aborting return to normal
```

```
* Description: Small Screen Method
*****************************************************************************/
METHOD returnToNormal_method
{
    LABEL [aborting_return_to_normal];
    DEFINITION
    {
        ACKNOWLEDGE(noteLoopReturned_string);
    }
}

/*****************************************************************************
* Method: Aborting return AO to normal
* Description: Small Screen Method
*****************************************************************************/
METHOD returnAOToNormal_method
{
    LABEL [aborting_return_to_normal];
    DEFINITION
    {
        char status[STATUS_SIZE];

        fassign(PV.DAQ.ANALOG_VALUE, 0.0);
        SEND_CMD(40)

        ACKNOWLEDGE(noteLoopReturned_string);
    }
}

/*****************************************************************************
* Method: Aborting return PV to normal
* Description: Auxiliary Abort Method
*****************************************************************************/
METHOD returnPVToNormal_method
{
    LABEL [aborting_return_to_normal];
    DEFINITION
    {
        char status[STATUS_SIZE];

        iassign(deviceVariables[device_variable_code].SIMULATED,
```

```
NORMAL_STATUS);
        SEND_CMD(79)

        ACKNOWLEDGE(noteLoopReturned_string);
    }
}

/*****************************************************************************
* Method: Lower Sensor Trim
* Description: Small Screen Method
*****************************************************************************/
METHOD lowerSensorTrim_method
{
    LABEL lowerSensorTrim_string;
    HELP lowerSensorTrim_help;
    DEFINITION
    {
        SENSOR_TRIM(LOW_TRIM_POINT)
    }
}

/*****************************************************************************
* Method: Upper Sensor Trim
* Description: Small Screen Method
*****************************************************************************/
METHOD upperSensorTrim_method
{
    LABEL upperSensorTrim_string;
    HELP upperSensorTrim_help;
    DEFINITION
    {
        SENSOR_TRIM(HIGH_TRIM_POINT)
    }
}

/*****************************************************************************
* Method: View Pressure Calibration Log
* Description: Large Screen Method
*****************************************************************************/
METHOD viewPressureCalibrationLog_method
{
```

```
    LABEL viewPressureCalibrationLog_string;
    HELP viewPressureCalibrationLog_help;
    DEFINITION
    {
        char status[STATUS_SIZE];
        int page;
        int letter;
        int indexArray;
        long select;
        select = 0;

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        for(page = 0; page < 4; page++)
        {
            iassign(pressureLogEntrySet, page);
            SEND_AND_FAIL_IF_ERROR(176)
            for(letter = 0; letter < 5; letter++)
            {
                indexArray = letter + (page * 5);
                pressureCalibrationLogDifference_array[indexArray] =
blank_string;
                pressureCalibrationLogFound_array[indexArray] =
blank_string;
                pressureCalibrationLogLeft_array[indexArray] =
blank_string;
                pressureCalibrationLogTimeSince_array[indexArray] =
blank_string;
                iassign(pressureCalibrationLogAction_array[indexArray],
actionValue_array[letter]);
                iassign(pressureCalibrationLogValSource_array[indexArray],
sourceValue_array[letter]);

                if(actionValue_array[letter] != NO_LOGGED_ACTION_ID)
                {
                    GET_STRING_BY_SECONDS(timeSinceValue_array[letter]);
                    pressureCalibrationLogTimeSince_array[indexArray] =
formattedDateTimeValue;
```

```
                    if((actionValue_array[letter] != LOG_CLEARED_ID) &&
(actionValue_array[letter] != RESTORE_FACTORY_TRIM_ID))
                    {
                        difference_local = abs(foundValue_array[letter] -
leftValue_array[letter]);
                        pressureCalibrationLogDifference_array[indexArray]
= BUILD_MESSAGE("%{difference_local} %{pressureUnits}");
                        found_local = foundValue_array[letter];
                        pressureCalibrationLogFound_array[indexArray] =
BUILD_MESSAGE("%{found_local} %{pressureUnits}");
                        left_local = leftValue_array[letter];
                        pressureCalibrationLogLeft_array[indexArray] =
BUILD_MESSAGE("%{left_local} %{pressureUnits}");
                    }
                }
            }
        }

        save_values();
        MenuDisplay(pressureCalibrationLog_dialog, next_string, select);
        process_abort();
    }
}

/*****************************************************************************
* Method: Clear Pressure Calibration Log
* Description: Small Screen Method
*****************************************************************************/
METHOD clearPressureCalibrationLog_method
{
    LABEL clearPressureCalibrationLog_string;
    HELP clearPressureCalibrationLog_help;
    DEFINITION
    {
        CLEAR_LOG(PRESSURE_CALIBRATION_LOG)
    }
}

/*****************************************************************************
* Method: Restore Pressure Calibration
* Description: Small Screen Method
```

```
*****************************************************************************/
METHOD restorePressureCalibration_method
{
    LABEL restorePressureCalibration_string;
    HELP restorePressureCalibration_help;
    DEFINITION
    {
        char status[STATUS_SIZE];

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        CHECK_LOCK_SW_SECURITY_SWITCH;

        ACKNOWLEDGE(warningLoopRemoved_string);
        add_abort_method(returnToNormal_method);

        device_variable_trim_code = PRESSURE_CODE;
        send(83, status);
        if(status[STATUS_RESPONSE_CODE] == 0)
        {
            SEND_AND_FAIL_IF_ERROR(80)
            ACKNOWLEDGE(pressureSensorCalibrationRestored_string);
        }
        else
        {
            display_response_status(83, status[STATUS_RESPONSE_CODE]);
        }

        process_abort();
    }
}

/*****************************************************************************
* Method: View New Configured Variables Log
* Description: Large Screen Method
*****************************************************************************/
METHOD viewNewConfiguredVarsLog_method
{
```

```
    LABEL viewconfiguredNewVarsLog_string;
    DEFINITION
    {
        char status[STATUS_SIZE];
        int letter;
        int indexArray;
        long select;
        select = 0;

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        SEND_AND_FAIL_IF_ERROR(175)
        for(letter = 0; letter < 10; letter++)
        {
            newVarsLogTimeSince_array[letter] = blank_string;
            iassign(newVarsLogEvents_array[letter],
actionValueNewVar_array[letter]);
            iassign(newVarsLogSource_array[letter],
sourceValueNewVar_array[letter]);

            if(actionValueNewVar_array[letter] != NO_LOGGED_ACTION_ID)
            {
                GET_STRING_BY_SECONDS(timeSinceValueNewVar_array[letter]);
                newVarsLogTimeSince_array[letter] =
formattedDateTimeValue;
            }
        }

        save_values();
        MenuDisplay(configuredNewVarsLog_dialog, next_string, select);
        process_abort();
    }
}

/*****************************************************************
* Method: Clear New Configured Variables Log
* Description: Small Screen Method
*****************************************************************/
```

```
METHOD clearNewConfiguredVariablesLog_method
{
    LABEL clearNewVarsLog_string;
    HELP clearNewVarLog_help;
    DEFINITION
    {
        CLEAR_LOG(VARIABLES_CONFIG_LOG)
    }
}

/*****************************************************************************
* Method: Blink Device
* Description: Small Screen Method
*****************************************************************************/
METHOD blinkDevice_method
{
    LABEL blinkDeviceMethod_string;
    HELP blinkDeviceMethod_help;
    DEFINITION
    {
        char status[STATUS_SIZE];

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        SEND_CMD(196)
        ACKNOWLEDGE(continingWillShow_string);
        iassign(squawk_control, SQUAWK_ONCE);

        do
        {
            SEND_AND_FAIL_IF_ERROR(72)
            DELAY(40, exitTheMethod_string);
        }
        while(1); //this loop should run until the user stops it

        process_abort();
    }
```

```
}

/*****************************************************************************
* Method: View 10 Most Recent Diagnostic Events
* Description: Large Screen Method
*****************************************************************************/
METHOD view10MostRecentDiagnosticEvents_method
{
    LABEL view10MostRecentDiagnosticEvents_string;
    HELP view10MostRecentDiagnosticEvents_help;
    DEFINITION
    {
        VIEW_DIAGNOSTIC_EVENTS(EVENT_10_PAGE)
    }
}

/*****************************************************************************
* Method: View 100 Most Recent Diagnostic Events
* Description: Large Screen Method
*****************************************************************************/
METHOD view100MostRecentDiagnosticEvents_method
{
    LABEL view100MostRecentDiagnosticEvents_string;
    HELP view100MostRecentDiagnosticEvents_help;
    DEFINITION
    {
        VIEW_DIAGNOSTIC_EVENTS(EVENT_100_PAGE)
    }
}

/*****************************************************************************
* Method: Clear Diagnostic Log
* Description: Small Screen Method
*****************************************************************************/
METHOD clearDiagnosticLog_method
{
    LABEL clearDiagnosticLog_string;
    HELP clearDiagnosticLog_help;
    DEFINITION
    {
        CLEAR_LOG(DIAGNOSTIC_LOG)
```

```
    }
}

/*****************************************************************************
* Method: Configure Loop Integrity
* Description: Small Screen Method
*****************************************************************************/
METHOD configureLoopIntegrity_method
{
    LABEL configureLoopIntegrity_string;
    HELP configureLoopIntegrity_help;
    DEFINITION
    {
        char status[STATUS_SIZE];
        int i;

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        CHECK_LOCK_SW_SECURITY_SWITCH;

        ACKNOWLEDGE(warningLoopRemoved_string);
        add_abort_method(returnToNormal_method);

        ACKNOWLEDGE(loopIntegrityDiagnosticMustPerform_string);
        SEND_AND_FAIL_IF_ERROR(191)
        DELAY(5, characterizationInProgress_string);

        for(i = 0; i < 4; i++)
        {
            SEND_AND_FAIL_IF_ERROR(193)
            if(operatingRegionLimits != CHARACTERIZATION_NOT_COMPLETE)
            {
                break;
            }
        }

        fassign(voltageDeviationLimit_local, voltageDeviationLimit);
        iassign(loopIntegrityNotificationMode_local,
```

```
loopIntegrityNotificationMode);
        save_values();

        switch(operatingRegionLimits)
        {
            case BELOW_LOWER_LIMIT:
                ACKNOWLEDGE(loopCharacterizationFailedBelowLimit_string);
                break;
            case CHARACTERIZATION_NOT_COMPLETE:
                ACKNOWLEDGE(loopCharacterizationFailedBelowLimit_string);
                break;
            case ABOVE_UPPER_LIMIT:
                ACKNOWLEDGE(loopCharacterizationFailedAboveLimit_string);
                break;
            default:
                GET_DEV_VAR_VALUE(enterVoltageDeviationLimit_string,
voltageDeviationLimit_local);
                GET_DEV_VAR_VALUE(selectDesiredNotificationMode_string,
loopIntegrityNotificationMode_local);
                iassign(loopIntegrityNotificationMode,
loopIntegrityNotificationMode_local);
                fassign(voltageDeviationLimit,
voltageDeviationLimit_local);
                SEND_AND_FAIL_IF_ERROR(192)
                ACKNOWLEDGE(configurationLoopDiagnosticComplete_string);
                break;
        }

        process_abort();
    }
}

#if __TOKVER__ >= 1000
/*****************************************************************
* Method: Get Health Status
* Description: Axillary Method For FDI
*****************************************************************/
/* This method is not directly referenced in the DD, but is used by
applications implementing FDI technology
to have a universal method of obtaining the health status from a field
device. */
```

```
METHOD GetHealthStatus
{
    LABEL getHealthStatus_string;
    TYPE unsigned char;
    DEFINITION
    {
        unsigned char returnValue;
        returnValue = HEALTH_STATUS_GOOD;
        if(FAILURE)
        {
            returnValue = HEALTH_STATUS_FAILURE;
        }
        else
        {
            if(MAINTENANCE_REQUIRED)
            {
                returnValue = HEALTH_STATUS_MAINTENANCE_REQIRED;
            }
            else
            {
                if(FUNCTION_CHECK)
                {
                    returnValue = HEALTH_STATUS_FUNCTION_CHECK;
                }
                else
                {
                    if(OUT_OF_SPECIFICATION)
                    {
                        returnValue = HEALTH_STATUS_OUT_OF_SPEC;
                    }
                }
            }
        }
        return returnValue;
    }
}
#endif

/*****************************************************************************
* Method: Lock / Unlock
* Description: Small Screen Method
```

```
*****************************************************************************/
METHOD lockUnlock_method
{
    LABEL lockUnlock_string;
    HELP lockUnlockHelp_string;
    DEFINITION
    {
        char status[STATUS_SIZE];

        IGNORE_ALL_DEVICE_STATUS();
        IGNORE_ALL_RESPONSE_CODES();
        XMTR_IGNORE_ALL_DEVICE_STATUS();
        XMTR_IGNORE_ALL_RESPONSE_CODES();

        SEND_AND_FAIL_IF_ERROR(76)
        if (lock_device_status_code == 0)
        {
            iassign(deviceLock_local, 0);
        }
        else
        {
            iassign(deviceLock_local, 1);
        }
        save_values();
        GET_DEV_VAR_VALUE(lockUnlockMethodText_string, deviceLock_local);

        if (deviceLock_local == 0)
        {
            iassign(lock_device_code, 0);
        }
        else
        {
            iassign(lock_device_code, 3);
        }
        SEND_AND_FAIL_IF_ERROR(71)
        // to update state on the screen, AMS does not do that for static
variables
        SEND_AND_FAIL_IF_ERROR(76)
        process_abort();
    }
}
```

Приложение Б

Листинг Б.1 – Исходный код команд

```
/*****************************************************************************
*   FILENAME: 997207_dev_commands.dd
*
*   DESCRIPTION: All device specific commands
*****************************************************************************/
COMMAND readStaticDataMaterials_command
{
    NUMBER 128;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            response_code,
            device_status,
            reserved,
            reserved,
            oringMaterialCode,
            reserved,
            reserved,
            reserved,
            reserved,
            reserved,
            reserved,
            moduleFillFluidCode,
            moduleIsolatorMaterialCode,
            reserved,
            reserved,
            pressureUnits,
            pressureLastUpperTrim,
            pressureLastLowerTrim,
            reserved,
            reservedEnabled,
            reserved,
            orderString
```

```
        }
    }
    RESPONSE_CODES
    {
        0, SUCCESS,               no_command_specific_errors_temp;
    }
}

COMMAND writeSegmentDisplayLanguage_command
{
    NUMBER 134;
    OPERATION WRITE;
    TRANSACTION
    {
        REQUEST
        {
            segmentDisplayLanguage
        }
        REPLY
        {
            response_code,
            device_status,
            segmentDisplayLanguage
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,     invalidSelectionDisplayLanguage_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
        7,  MODE_ERROR,           in_write_protect_mode_temp;
    }
}

COMMAND writeDisplayBacklight_command
{
    NUMBER 135;
    OPERATION WRITE;
    TRANSACTION
    {
```

```
        REQUEST
        {
            backlight
        }
        REPLY
        {
            response_code,
            device_status,
            backlight
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,     invalidSelectionDisplayLanguage_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
        7,  MODE_ERROR,           in_write_protect_mode_temp;
        16, MODE_ERROR,           accessRestricted_string;
    }
}

COMMAND writeDisplayParameters_command
{
    NUMBER 136;
    OPERATION WRITE;
    TRANSACTION
    {
        REQUEST
        {
            displayParameters
        }
        REPLY
        {
            response_code,
            device_status,
            displayParameters
        }
    }
    RESPONSE_CODES
    {
```

```
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,     invalidSelectionDisplayParameters_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
        7,  MODE_ERROR,           in_write_protect_mode_temp;
        16, MODE_ERROR,           accessRestricted_string;
    }
}

COMMAND readDisplayConfiguration_command
{
    NUMBER 137;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            response_code,
            device_status,
            segmentDisplayLanguage,
            displayParameters,
            backlight
        }
    }
    RESPONSE_CODES
    {
        0, SUCCESS,               no_command_specific_errors_temp;
    }
}

COMMAND writeBacklightBlinkingStatuses_command
{
    NUMBER 138;
    OPERATION WRITE;
    TRANSACTION
    {
        REQUEST
        {
```

```
            statusMaskForBacklight0,
            statusMaskForBacklight1,
            statusMaskForBacklight2,
            statusMaskForBacklight3,
            statusMaskForBacklight4,
            statusMaskForBacklight5
        }
        REPLY
        {
            response_code,
            device_status,
            statusMaskForBacklight0,
            statusMaskForBacklight1,
            statusMaskForBacklight2,
            statusMaskForBacklight3,
            statusMaskForBacklight4,
            statusMaskForBacklight5
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
        7,  MODE_ERROR,           in_write_protect_mode_temp;
        16, MODE_ERROR,           accessRestricted_string;
    }
}

COMMAND readBacklightBlinkingStatuses_command
{
    NUMBER 139;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            response_code,
```

```
            device_status,
            statusMaskForBacklight0,
            statusMaskForBacklight1,
            statusMaskForBacklight2,
            statusMaskForBacklight3,
            statusMaskForBacklight4,
            statusMaskForBacklight5
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
    }
}

COMMAND writeAlarmAndSaturationValues_command
{
    NUMBER 140;
    OPERATION WRITE;
    TRANSACTION 0
    {
        REQUEST
        {
            alarmSaturationSetting,
            methodHighAlarm,
            methodLowAlarm,
            methodHighSaturation,
            methodLowSaturation
        }
        REPLY
        {
            response_code,
            device_status,
            alarmSaturationSetting,
            methodHighAlarm,
            methodLowAlarm,
            methodHighSaturation,
            methodLowSaturation
        }
    }
    TRANSACTION 1
```

```
    {
        REQUEST
        {
            ALARM_SATURATION_CUSTOM,
            screenHighAlarm,
            screenLowAlarm,
            screenHighSaturation,
            screenLowSaturation
        }
        REPLY
        {
            response_code,
            device_status,
            ALARM_SATURATION_CUSTOM,
            screenHighAlarm,
            screenLowAlarm,
            screenHighSaturation,
            screenLowSaturation
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,     invalidSelectionAlarmSaturationRC_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
        7,  MODE_ERROR,           in_write_protect_mode_temp;
        9,  DATA_ENTRY_ERROR,     lowAlarmValueTooHigh_string;
        10, DATA_ENTRY_ERROR,     lowAlarmValueTooLow_string;
        11, DATA_ENTRY_ERROR,     highAlarmValueTooHigh_string;
        12, DATA_ENTRY_ERROR,     highAlarmValueTooLow_string;
        13, DATA_ENTRY_ERROR,     lowSaturationValueToHigh_string;
        15, DATA_ENTRY_ERROR,     lowSaturationValueToLow_string;
        16, MODE_ERROR,           accessRestricted_string;
        28, DATA_ENTRY_ERROR,     highSaturationValueToHigh_string;
        29, DATA_ENTRY_ERROR,     highSaturationValueToLow_string;
    }
}

COMMAND performAOFactoryTrimRecall_command
{
```

```
    NUMBER 141;
    OPERATION COMMAND;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            response_code,
            device_status
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
        7,  MODE_ERROR,           in_write_protect_mode_temp;
        16, MODE_ERROR,           accessRestricted_string;
    }
}

COMMAND readAlarmAndSaturationValues_command
{
    NUMBER 142;
    OPERATION READ;
    TRANSACTION 0
    {
        REQUEST
        {
        }
        REPLY
        {
            response_code,
            device_status,
            alarmSaturationSetting,
            methodHighAlarm,
            methodLowAlarm,
            methodHighSaturation,
            methodLowSaturation
        }
```

```
        }
        TRANSACTION 1
        {
            REQUEST
            {
            }
            REPLY
            {
                response_code,
                device_status,
                alarmSaturationSetting,
                screenHighAlarm,
                screenLowAlarm,
                screenHighSaturation,
                screenLowSaturation
            }

        }
        RESPONSE_CODES
        {
            0,  SUCCESS,              no_command_specific_errors_temp;
        }
    }

    // FIXME: This command commented since it's decided to hide Namur HW/SW
Revision.
    // COMMAND readNamurRevisions_command
    // {
    //     NUMBER 143;
    //     OPERATION READ;
    //     TRANSACTION 0
    //     {
    //         REQUEST
    //         {
    //         }
    //         REPLY
    //         {
    //             response_code,
    //             device_status,
    //             namurHardwareRevision,
    //             namurSoftwareRevision
```

```
//         }
//     }
//     RESPONSE_CODES
//     {
//         0,  SUCCESS,              no_command_specific_errors_temp;
//     }
// }

COMMAND readTaggingInformation_command
{
    NUMBER 145;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
            tModelTaggingChoice_local (INFO,INDEX)
        }
        REPLY
        {
            response_code,
            device_status,
            tModelTaggingChoice_local (INFO,INDEX),
            modelTaggingInfo_array[tModelTaggingChoice_local]
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,    invalidSelectionModelTaggingChoice_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
    }
}

COMMAND readModelAndSerialNumbers_command
{
    NUMBER 146;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
```

```
        {
        }
        REPLY
        {
            response_code,
            device_status,
            modelNumber
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
    }
}

COMMAND writeFlowSetup_command
{
    NUMBER 148;
    OPERATION WRITE;
    TRANSACTION
    {
        REQUEST
        {
            flowSetupPressureInput,
            flowSetupFlowInput,
            flowUnitsAscii
        }
        REPLY
        {
            response_code,
            device_status,
            flowSetupPressureInput,
            flowSetupFlowInput,
            flowUnitsAscii
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
```

```
        7,  MODE_ERROR,           in_write_protect_mode_temp;
        9,  DATA_ENTRY_ERROR,     pressureValueTooLarge_string;
       10,  DATA_ENTRY_ERROR,     pressureValueTooSmall_string;
       11,  DATA_ENTRY_ERROR,     flowValueTooLarge_string;
       12,  DATA_ENTRY_ERROR,     flowValueTooSmall_string;
       16,  MODE_ERROR,           accessRestricted_string;
    }
}

COMMAND writeLowFlowCuttoffConfiguration_command
{
    NUMBER 149;
    OPERATION WRITE;
    TRANSACTION
    {
        REQUEST
        {
            lowFlowCutoffMode,
            lowFlowCutoffPressure,
            lowFlowCutinPressure
        }
        REPLY
        {
            response_code,
            device_status,
            lowFlowCutoffMode,
            lowFlowCutoffPressure,
            lowFlowCutinPressure
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,     invalidSelectionLowFlowCutoffMode_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
        7,  MODE_ERROR,           in_write_protect_mode_temp;
        9,  DATA_ENTRY_ERROR,     cutoffValueTooHigh_string;
       10,  DATA_ENTRY_ERROR,     cutoffValueTooLow_string;
       11,  DATA_ENTRY_ERROR,     cutinValueTooHigh_string;
       12,  DATA_ENTRY_ERROR,     cutinValueTooLow_string;
```

```
            13,                                                DATA_ENTRY_ERROR,
cutoffValueMustBeLessThanCutinValue_string;
            16,  MODE_ERROR,           accessRestricted_string;
        }
    }


    COMMAND readFlowSetup_command
    {
        NUMBER 150;
        OPERATION READ;
        TRANSACTION
        {
            REQUEST
            {
            }
            REPLY
            {
                response_code,
                device_status,
                pressureUnits,
                flowSetupPressureInput,
                flowSetupFlowInput,
                flowUnitsAscii,
                lowFlowCutoffMode,
                lowFlowCutoffPressure,
                lowFlowCutinPressure
            }
        }
        RESPONSE_CODES
        {
            0,  SUCCESS,              no_command_specific_errors_temp;
        }
    }

    COMMAND writeTotalizerSetup_command
    {
        NUMBER 151;
        OPERATION WRITE;
        TRANSACTION
        {
```

```
        REQUEST
        {
            totalizerUnitsAscii,
            totalizerFlowTimeUnit,
            totalizerUnitConversionFactor,
            totalizerDirection
        }
        REPLY
        {
            response_code,
            device_status,
            totalizerUnitsAscii,
            totalizerFlowTimeUnit,
            totalizerUnitConversionFactor,
            totalizerDirection
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,     invalidSelectionTotalizerSetup_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
        7,  MODE_ERROR,           in_write_protect_mode_temp;
        9,  DATA_ENTRY_ERROR,     totalizerUnitConversionTooHigh_string;
        10, DATA_ENTRY_ERROR,     totalizerUnitConversionTooLow_string;
        16, MODE_ERROR,           accessRestricted_string;
    }
}

COMMAND writeTotalizerMode_command
{
    NUMBER 152;
    OPERATION WRITE;
    TRANSACTION
    {
        REQUEST
        {
            totalizerMode
        }
        REPLY
```

```
        {
            response_code,
            device_status,
            totalizerMode
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,     invalidSelectionTotalizerMode_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
        7,  MODE_ERROR,           in_write_protect_mode_temp;
        16, MODE_ERROR,           accessRestricted_string;
    }
}

COMMAND readTotalizerSetup_command
{
    NUMBER 153;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            response_code,
            device_status,
            totalizerMode,
            totalizerUnitsAscii,
            totalizerFlowTimeUnit,
            totalizerUnitConversionFactor,
            totalizerDirection,
            totalizerUsl,
            flowUnitsAscii
        }
    }
    RESPONSE_CODES
    {
```

```
        0,  SUCCESS,              no_command_specific_errors_temp;
    }
}

COMMAND setOptionToZero_command
{
    NUMBER 154;
    OPERATION COMMAND;
    TRANSACTION
    {
        REQUEST
        {
            tSetToZeroOption (INFO)
        }
        REPLY
        {
            response_code,
            device_status,
            tSetToZeroOption (INFO)
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,     invalidSelectionSetOptionToZero_string;
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
        7,  MODE_ERROR,           in_write_protect_mode_temp;
        16, MODE_ERROR,           accessRestricted_string;
    }
}

COMMAND writeLevelSetup_command
{
    NUMBER 155;
    OPERATION WRITE;
    TRANSACTION 0
    {
        REQUEST
        {
            writeLevelUnits (INFO),
            writePressureUnits (INFO),
```

```
        methodLevelSetupMaxLevel,
        methodLevelSetupPressureAtMaxLevel,
        methodLevelSetupMinLevel,
        methodLevelSetupPressureAtMinLevel
    }
    REPLY
    {
        response_code,
        device_status,
        writeLevelUnits (INFO),
        writePressureUnits (INFO),
        methodLevelSetupMaxLevel,
        methodLevelSetupPressureAtMaxLevel,
        methodLevelSetupMinLevel,
        methodLevelSetupPressureAtMinLevel
    }
}
TRANSACTION 1
{
    REQUEST
    {
        levelConfigUnits (INFO),
        pressureConfigUnits (INFO),
        screenLevelSetupMaxLevel,
        screenLevelSetupPressureAtMaxLevel,
        screenLevelSetupMinLevel,
        screenLevelSetupPressureAtMinLevel
    }
    REPLY
    {
        response_code,
        device_status,
        levelConfigUnits (INFO),
        pressureConfigUnits (INFO),
        screenLevelSetupMaxLevel,
        screenLevelSetupPressureAtMaxLevel,
        screenLevelSetupMinLevel,
        screenLevelSetupPressureAtMinLevel
    }
}
RESPONSE_CODES
```

```
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,     invalidSelectionUnitsCode_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
        7,  MODE_ERROR,           in_write_protect_mode_temp;
        9,  DATA_ENTRY_ERROR,     maxLevelTooHigh_string;
        10, DATA_ENTRY_ERROR,     maxLevelTooLow_string;
        11, DATA_ENTRY_ERROR,     minLevelTooHigh_string;
        12, DATA_ENTRY_ERROR,     minLevelTooLow_string;
        13, DATA_ENTRY_ERROR,     pressureAtMaxLevelTooHigh_string;
        15, DATA_ENTRY_ERROR,     pressureAtMaxLevelTooLow_string;
        16, MODE_ERROR,           accessRestricted_string;
        28, DATA_ENTRY_ERROR,     pressureAtMinLevelTooHigh_string;
        29, DATA_ENTRY_ERROR,     pressureAtMinLevelTooLow_string;
        65, DATA_ENTRY_ERROR,     minLevelMustBeLessThanMaxLevel_string;
    }
}


COMMAND adjustLevelReading_command
{
    NUMBER 156;
    OPERATION COMMAND;
    TRANSACTION
    {
        REQUEST
        {
            actualLevel (INFO)
        }
        REPLY
        {
            response_code,
            device_status,
            actualLevel (INFO)
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
```

```
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
        7,  MODE_ERROR,           in_write_protect_mode_temp;
        9,  DATA_ENTRY_ERROR,     applied_process_too_high_temp;
        10, DATA_ENTRY_ERROR,     applied_process_too_low_temp;
        16, MODE_ERROR,           accessRestricted_string;
    }
}

COMMAND readLevelConfiguration_command
{
    NUMBER 157;
    OPERATION READ;
    TRANSACTION 0
    {
        REQUEST
        {
        }
        REPLY
        {
            response_code,
            device_status,
            levelUnits,
            pressureUnits,
            methodLevelSetupMaxLevel,
            methodLevelSetupPressureAtMaxLevel,
            methodLevelSetupMinLevel,
            methodLevelSetupPressureAtMinLevel,
            levelUsl,
            levelLsl,
            levelAdjustment
        }
    }
    TRANSACTION 1
    {
        REQUEST
        {
        }
        REPLY
        {
            response_code,
            device_status,
```

```
            levelUnits,
            pressureUnits,
            screenLevelSetupMaxLevel,
            screenLevelSetupPressureAtMaxLevel,
            screenLevelSetupMinLevel,
            screenLevelSetupPressureAtMinLevel,
            levelUsl,
            levelLsl,
            levelAdjustment
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
    }
}


COMMAND configureVolumeSetup_command
{
    NUMBER 158;
    OPERATION COMMAND;
    TRANSACTION
    {
        REQUEST
        {
            volumeSetupNumStrapWritePoints,
            volumeSetupTankWriteType,
            volumeSetupTankWriteLength,
            volumeSetupTankWriteRadius
        }
        REPLY
        {
            response_code,
            device_status,
            volumeSetupNumStrapWritePoints,
            volumeSetupTankWriteType,
            volumeSetupTankWriteLength,
            volumeSetupTankWriteRadius
        }
    }
```

```
        RESPONSE_CODES
        {
            0,  SUCCESS,              no_command_specific_errors_temp;
            2,                                                DATA_ENTRY_ERROR,
invalidSelectionConfigureVolumeSetup_string;
            5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
            6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
            7,  MODE_ERROR,           in_write_protect_mode_temp;
            9,  DATA_ENTRY_ERROR,     tooManyStrappingPoints_string;
            10, DATA_ENTRY_ERROR,     tooFewStrappingPoints_string;
            16, MODE_ERROR,           accessRestricted_string;
        }
    }

    COMMAND configureVolumeStrappingTable_command
    {
        NUMBER 159;
        OPERATION COMMAND;
        TRANSACTION
        {
            REQUEST
            {
                writeStrappingPointSet_local (INFO,INDEX),
                levelConfigUnitsCommand (INFO),
                volumeConfigUnitsCommand (INFO),

writeValueStrappingPointSet_array[writeStrappingPointSet_local].LEVELA,

writeValueStrappingPointSet_array[writeStrappingPointSet_local].LEVELB,

writeValueStrappingPointSet_array[writeStrappingPointSet_local].LEVELC,

writeValueStrappingPointSet_array[writeStrappingPointSet_local].LEVELD,

writeValueStrappingPointSet_array[writeStrappingPointSet_local].VOLUMEA,

writeValueStrappingPointSet_array[writeStrappingPointSet_local].VOLUMEB,

writeValueStrappingPointSet_array[writeStrappingPointSet_local].VOLUMEC,

writeValueStrappingPointSet_array[writeStrappingPointSet_local].VOLUMED
```

```
            }
            REPLY
            {
                response_code,
                device_status,
                writeStrappingPointSet_local (INFO,INDEX),
                levelConfigUnitsCommand (INFO),
                volumeConfigUnitsCommand (INFO),

writeValueStrappingPointSet_array[writeStrappingPointSet_local].LEVELA,

writeValueStrappingPointSet_array[writeStrappingPointSet_local].LEVELB,

writeValueStrappingPointSet_array[writeStrappingPointSet_local].LEVELC,

writeValueStrappingPointSet_array[writeStrappingPointSet_local].LEVELD,

writeValueStrappingPointSet_array[writeStrappingPointSet_local].VOLUMEA,

writeValueStrappingPointSet_array[writeStrappingPointSet_local].VOLUMEB,

writeValueStrappingPointSet_array[writeStrappingPointSet_local].VOLUMEC,

writeValueStrappingPointSet_array[writeStrappingPointSet_local].VOLUMED
            }
        }
        RESPONSE_CODES
        {
            0,  SUCCESS,              no_command_specific_errors_temp;
            2,  DATA_ENTRY_ERROR,     invalidSelectionStrappingPointSet_string;
            5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
            6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
            7,  MODE_ERROR,           in_write_protect_mode_temp;
            9,  DATA_ENTRY_ERROR,     numberOfStrappingPointsNotDefined_string;
            10, DATA_ENTRY_ERROR,     strappingPointSetOutOfOrder_string;
            11, DATA_ENTRY_ERROR,     levelValueAIsNotMonotonic_string;
            12, DATA_ENTRY_ERROR,     levelValueBIsNotMonotonic_string;
            13, DATA_ENTRY_ERROR,     levelValueCIsNotMonotonic_string;
            15, DATA_ENTRY_ERROR,     levelValueDIsNotMonotonic_string;
            16, MODE_ERROR,           accessRestricted_string;
            28, DATA_ENTRY_ERROR,     volumeValueAIsNotMonotonic_string;
```

```
        29, DATA_ENTRY_ERROR,     volumeValueBIsNotMonotonic_string;
        65, DATA_ENTRY_ERROR,     volumeValueCIsNotMonotonic_string;
        66, DATA_ENTRY_ERROR,     volumeValueDIsNotMonotonic_string;
    }
}

COMMAND activateVolumeConfiguration_command
{
    NUMBER 160;
    OPERATION COMMAND;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            response_code,
            device_status
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
        7,  MODE_ERROR,           in_write_protect_mode_temp;
        9,  DATA_ENTRY_ERROR,     incompleteStrappingTable_string;
        16, MODE_ERROR,           accessRestricted_string;
    }
}

COMMAND readVolumeSetup_command
{
    NUMBER 161;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
```

```
            {
                response_code,
                device_status,
                levelUnits,
                volumeUnits,
                volumeSetupNumStrapReadPoints,
                volumeSetupTankReadType,
                volumeSetupTankReadLength,
                volumeSetupTankReadRadius,
                volumeUsl
            }
        }
        RESPONSE_CODES
        {
            0,  SUCCESS,              no_command_specific_errors_temp;
        }
    }

    COMMAND readVolumeStrappingTable_command
    {
        NUMBER 162;
        OPERATION READ;
        TRANSACTION
        {
            REQUEST
            {
                readStrappingPointSet_local (INFO,INDEX)
            }
            REPLY
            {
                response_code,
                device_status,
                readStrappingPointSet_local (INFO,INDEX),
                levelUnits,
                volumeUnits,

readValueStrappingPointSet_array[readStrappingPointSet_local].LEVELA,

readValueStrappingPointSet_array[readStrappingPointSet_local].LEVELB,

readValueStrappingPointSet_array[readStrappingPointSet_local].LEVELC,
```

```
readValueStrappingPointSet_array[readStrappingPointSet_local].LEVELD,

readValueStrappingPointSet_array[readStrappingPointSet_local].LEVELE,

readValueStrappingPointSet_array[readStrappingPointSet_local].LEVELF,

readValueStrappingPointSet_array[readStrappingPointSet_local].LEVELG,

readValueStrappingPointSet_array[readStrappingPointSet_local].LEVELH,

readValueStrappingPointSet_array[readStrappingPointSet_local].LEVELI,

readValueStrappingPointSet_array[readStrappingPointSet_local].LEVELJ,

readValueStrappingPointSet_array[readStrappingPointSet_local].VOLUMEA,

readValueStrappingPointSet_array[readStrappingPointSet_local].VOLUMEB,

readValueStrappingPointSet_array[readStrappingPointSet_local].VOLUMEC,

readValueStrappingPointSet_array[readStrappingPointSet_local].VOLUMED,

readValueStrappingPointSet_array[readStrappingPointSet_local].VOLUMEE,

readValueStrappingPointSet_array[readStrappingPointSet_local].VOLUMEF,

readValueStrappingPointSet_array[readStrappingPointSet_local].VOLUMEG,

readValueStrappingPointSet_array[readStrappingPointSet_local].VOLUMEH,

readValueStrappingPointSet_array[readStrappingPointSet_local].VOLUMEI,

readValueStrappingPointSet_array[readStrappingPointSet_local].VOLUMEJ
            }
        }
        RESPONSE_CODES
        {
            0,  SUCCESS,              no_command_specific_errors_temp;
            2,  DATA_ENTRY_ERROR,     invalidSelectionStrappingPointSet_string;
            5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
```

```
    }
}

COMMAND writeAlarmDirection_command
{
    NUMBER 164;
    OPERATION WRITE;
    TRANSACTION
    {
        REQUEST
        {
            loop_alarm_code
        }
        REPLY
        {
            response_code,
            device_status,
            loop_alarm_code
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,     invalidSelectionAlarmDirection_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
        7,  MODE_ERROR,           in_write_protect_mode_temp;
        16, MODE_ERROR,           accessRestricted_string;
    }
}

COMMAND writeSoftwareSecurity_command
{
    NUMBER 165;
    OPERATION WRITE;
    TRANSACTION
    {
        REQUEST
        {
            softwareSecurity
        }
```

```
        REPLY
        {
            response_code,
            device_status,
            softwareSecurity
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,     invalidSelectionSoftwareSecurity_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
    }
}

COMMAND readDiagnosticLog_command
{
    NUMBER 168;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
            diagnosticLogEntrySet (INFO)
        }
        REPLY
        {
            response_code,
            device_status,
            diagnosticLogEntrySet (INFO),
            eventValue_array[A],
            timeSinceValueDiagnostic_array[A],
            eventValue_array[B],
            timeSinceValueDiagnostic_array[B],
            eventValue_array[C],
            timeSinceValueDiagnostic_array[C],
            eventValue_array[D],
            timeSinceValueDiagnostic_array[D],
            eventValue_array[E],
            timeSinceValueDiagnostic_array[E],
```

```
            eventValue_array[F],
            timeSinceValueDiagnostic_array[F],
            eventValue_array[G],
            timeSinceValueDiagnostic_array[G],
            eventValue_array[H],
            timeSinceValueDiagnostic_array[H],
            eventValue_array[I],
            timeSinceValueDiagnostic_array[I],
            eventValue_array[J],
            timeSinceValueDiagnostic_array[J]
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,     invalidSelectionDiagnosticLog_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
    }
}

COMMAND readNumberOfDiagnosticLogEntries_command
{
    NUMBER 169;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            response_code,
            device_status,
            diagLogNumberOfEvents
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
    }
}
```

```
COMMAND readMinMaxValuesLog_command
{
    NUMBER 173;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
            minMaxLogType_local (INFO,INDEX)
        }
        REPLY
        {
            response_code,
            device_status,
            minMaxLogType_local (INFO,INDEX),
            minMaxLog_array[minMaxLogType_local].MIN_VALUE_SEEN,
            minMaxLog_array[minMaxLogType_local].MIN_TIME_SINCE,
            minMaxLog_array[minMaxLogType_local].MAX_VALUE_SEEN,
            minMaxLog_array[minMaxLogType_local].MAX_TIME_SINCE
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,     invalidSelectionDiagnosticLog_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
    }
}

COMMAND clearLog_command
{
    NUMBER 174;
    OPERATION COMMAND;
    TRANSACTION
    {
        REQUEST
        {
            tLogToClear (INFO)
        }
        REPLY
        {
```

```
            response_code,
            device_status,
            tLogToClear (INFO)
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,     invalidSelectionClearLog_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
        7,  MODE_ERROR,           in_write_protect_mode_temp;
        16, MODE_ERROR,           accessRestricted_string;
    }
}

COMMAND readVariablesConfigurationLog_command
{
    NUMBER 175;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            response_code,
            device_status,
            actionValueNewVar_array[0],
            timeSinceValueNewVar_array[0],
            sourceValueNewVar_array[0],
            actionValueNewVar_array[1],
            timeSinceValueNewVar_array[1],
            sourceValueNewVar_array[1],
            actionValueNewVar_array[2],
            timeSinceValueNewVar_array[2],
            sourceValueNewVar_array[2],
            actionValueNewVar_array[3],
            timeSinceValueNewVar_array[3],
            sourceValueNewVar_array[3],
```

```
            actionValueNewVar_array[4],
            timeSinceValueNewVar_array[4],
            sourceValueNewVar_array[4],
            actionValueNewVar_array[5],
            timeSinceValueNewVar_array[5],
            sourceValueNewVar_array[5],
            actionValueNewVar_array[6],
            timeSinceValueNewVar_array[6],
            sourceValueNewVar_array[6],
            actionValueNewVar_array[7],
            timeSinceValueNewVar_array[7],
            sourceValueNewVar_array[7],
            actionValueNewVar_array[8],
            timeSinceValueNewVar_array[8],
            sourceValueNewVar_array[8],
            actionValueNewVar_array[9],
            timeSinceValueNewVar_array[9],
            sourceValueNewVar_array[9]
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
    }
}

COMMAND readPressureCalibrationLog_command
{
    NUMBER 176;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
            pressureLogEntrySet (INFO)
        }
        REPLY
        {
            response_code,
            device_status,
            pressureLogEntrySet (INFO),
```

```
            actionValue_array[A],
            timeSinceValue_array[A],
            sourceValue_array[A],
            foundValue_array[A],
            leftValue_array[A],
            pressureUnitsValue_array[A],
            actionValue_array[B],
            timeSinceValue_array[B],
            sourceValue_array[B],
            foundValue_array[B],
            leftValue_array[B],
            pressureUnitsValue_array[B],
            actionValue_array[C],
            timeSinceValue_array[C],
            sourceValue_array[C],
            foundValue_array[C],
            leftValue_array[C],
            pressureUnitsValue_array[C],
            actionValue_array[D],
            timeSinceValue_array[D],
            sourceValue_array[D],
            foundValue_array[D],
            leftValue_array[D],
            pressureUnitsValue_array[D],
            actionValue_array[E],
            timeSinceValue_array[E],
            sourceValue_array[E],
            foundValue_array[E],
            leftValue_array[E],
            pressureUnitsValue_array[E]
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,     invalidSelectionCalibrationLog_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
    }
}

COMMAND readAnalogOutputCalibrationLog_command
```

```
{
    NUMBER 177;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
            aoLogEntrySet (INFO)
        }
        REPLY
        {
            response_code,
            device_status,
            aoLogEntrySet (INFO),
            actionValueAO_array[A],
            timeSinceValueAO_array[A],
            sourceValueAO_array[A],
            foundValueAO_array[A],
            leftValueAO_array[A],
            foundDevVarValueAO_array[A],
            leftDevVarValueAO_array[A],
            unitsCodeValueAO_array[A],
            actionValueAO_array[B],
            timeSinceValueAO_array[B],
            sourceValueAO_array[B],
            foundValueAO_array[B],
            leftValueAO_array[B],
            foundDevVarValueAO_array[B],
            leftDevVarValueAO_array[B],
            unitsCodeValueAO_array[B],
            actionValueAO_array[C],
            timeSinceValueAO_array[C],
            sourceValueAO_array[C],
            foundValueAO_array[C],
            leftValueAO_array[C],
            foundDevVarValueAO_array[C],
            leftDevVarValueAO_array[C],
            unitsCodeValueAO_array[C],
            actionValueAO_array[D],
            timeSinceValueAO_array[D],
            sourceValueAO_array[D],
```

```
            foundValueAO_array[D],
            leftValueAO_array[D],
            foundDevVarValueAO_array[D],
            leftDevVarValueAO_array[D],
            unitsCodeValueAO_array[D],
            actionValueAO_array[E],
            timeSinceValueAO_array[E],
            sourceValueAO_array[E],
            foundValueAO_array[E],
            leftValueAO_array[E],
            foundDevVarValueAO_array[E],
            leftDevVarValueAO_array[E],
            unitsCodeValueAO_array[E]
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,     invalidSelectionCalibrationLog_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
    }
}

COMMAND setLOIPasswordProtection_command
{
    NUMBER 190;
    OPERATION WRITE;
    TRANSACTION
    {
        REQUEST
        {
            loiPasswordProtection,
            loiPassword
        }
        REPLY
        {
            response_code,
            device_status,
            loiPasswordProtection,
            loiPassword
        }
```

```
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,    invalidSelectionPasswordProtection_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
        7,  MODE_ERROR,           in_write_protect_mode_temp;
        16, MODE_ERROR,           accessRestricted_string;
        65, DATA_ENTRY_ERROR,     invalidPassword_string;
    }
}

COMMAND characterizeLoop_command
{
    NUMBER 191;
    OPERATION COMMAND;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            response_code,
            device_status
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        7,  MODE_ERROR,           in_write_protect_mode_temp;
        9,  DATA_ENTRY_ERROR,     loopCharacterizationInProgress_string;
        11, DATA_ENTRY_ERROR,     loopCurrentDisabled_string;
        16, MODE_ERROR,           accessRestricted_string;
        70, MISC_ERROR,           featureNotAvailable_string;
    }
}

COMMAND writeLoopIntegrityConfiguration_command
{
```

```
    NUMBER 192;
    OPERATION WRITE;
    TRANSACTION
    {
        REQUEST
        {
            loopIntegrityNotificationMode,
            voltageDeviationLimit
        }
        REPLY
        {
            response_code,
            device_status,
            loopIntegrityNotificationMode,
            voltageDeviationLimit
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,     invalidSelectionLoopIntegrity_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
        6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
        7,  MODE_ERROR,           in_write_protect_mode_temp;
        9,  DATA_ENTRY_ERROR,     voltageDeviationTooHigh_string;
        10, DATA_ENTRY_ERROR,     voltageDeviationTooLow_string;
        16, MODE_ERROR,           accessRestricted_string;
        112, DATA_ENTRY_WARNING,  featureNotAvailable_string;
        113, DATA_ENTRY_WARNING,  loopHasNotBeenCharacterized_string;
    }
}

COMMAND readLoopIntegrityConfiguration_command
{
    NUMBER 193;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
```

```
        REPLY
        {
            response_code,
            device_status,
            loopIntegrityNotificationMode,
            loopResistanceBaseline,
            powerSupplyVoltageBaseline,
            loopResistancePreviousBaseline,
            powerSupplyVoltagePreviousBaseline,
            minAllowableVoltageDeviation,
            maxAllowableVoltageDeviation,
            voltageDeviationLimit,
            operatingRegionLimits
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        70, MISC_ERROR,           featureNotAvailable_string;
    }
}

COMMAND readTerminalVoltage_command
{
    NUMBER 194;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            response_code,
            device_status,
            terminalVoltage
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
```

```
        70, MISC_ERROR,           featureNotAvailable_string;
    }
}

COMMAND readSecurityAndTransmitterOptions_command
{
    NUMBER 196;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
        }
        REPLY
        {
            response_code,
            device_status,
            softwareSecurity,
            lock_device_status_code,
            loiPasswordProtection,
            powerAdvisoryEnabled
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
    }
}

COMMAND writeProcessAlertSetup_command
{
    NUMBER 200;
    OPERATION WRITE;
    TRANSACTION 0
    {
        REQUEST
        {
            0,
            processAlert1Name,
            processAlert1NotificationMode,
            processAlert1MonitoredDeviceVar,
```

```
        processAlert1ActivationTrigger,
        processAlert1HighAlertValue,
        processAlert1LowAlertValue,
        processAlert1Suppression,
        processAlert1TimeDelay,
        processAlert1Deadband
    }
    REPLY
    {
        response_code,
        device_status,
        0,
        processAlert1Name,
        processAlert1NotificationMode,
        processAlert1MonitoredDeviceVar,
        processAlert1ActivationTrigger,
        processAlert1HighAlertValue,
        processAlert1LowAlertValue,
        processAlert1Suppression,
        processAlert1TimeDelay,
        processAlert1Deadband
    }
}
TRANSACTION 1
{
    REQUEST
    {
        1,
        processAlert2Name,
        processAlert2NotificationMode,
        processAlert2MonitoredDeviceVar,
        processAlert2ActivationTrigger,
        processAlert2HighAlertValue,
        processAlert2LowAlertValue,
        processAlert2Suppression,
        processAlert2TimeDelay,
        processAlert2Deadband
    }
    REPLY
    {
        response_code,
```

```
                device_status,
                1,
                processAlert2Name,
                processAlert2NotificationMode,
                processAlert2MonitoredDeviceVar,
                processAlert2ActivationTrigger,
                processAlert2HighAlertValue,
                processAlert2LowAlertValue,
                processAlert2Suppression,
                processAlert2TimeDelay,
                processAlert2Deadband
            }
        }
        TRANSACTION 2
        {
            REQUEST
            {
                tProcessAlert_local (INFO,INDEX),
                processAlerts_array[tProcessAlert_local].OFFLINE_NAME,

processAlerts_array[tProcessAlert_local].OFFLINE_NOTIFICATION_MODE,

processAlerts_array[tProcessAlert_local].OFFLINE_DEVICE_VARIABLE,

processAlerts_array[tProcessAlert_local].OFFLINE_ACTIVATION_TRIGGER,
                processAlerts_array[tProcessAlert_local].OFFLINE_HIGH_VALUE,
                processAlerts_array[tProcessAlert_local].OFFLINE_LOW_VALUE,
                processAlerts_array[tProcessAlert_local].OFFLINE_SUPPRESSION,
                processAlerts_array[tProcessAlert_local].OFFLINE_TIME_DELAY,
                processAlerts_array[tProcessAlert_local].OFFLINE_DEADBAND
            }
            REPLY
            {
                response_code,
                device_status,
                tProcessAlert_local (INFO,INDEX),
                processAlerts_array[tProcessAlert_local].OFFLINE_NAME,

processAlerts_array[tProcessAlert_local].OFFLINE_NOTIFICATION_MODE,

processAlerts_array[tProcessAlert_local].OFFLINE_DEVICE_VARIABLE,
```

```
processAlerts_array[tProcessAlert_local].OFFLINE_ACTIVATION_TRIGGER,
                processAlerts_array[tProcessAlert_local].OFFLINE_HIGH_VALUE,
                processAlerts_array[tProcessAlert_local].OFFLINE_LOW_VALUE,
                processAlerts_array[tProcessAlert_local].OFFLINE_SUPPRESSION,
                processAlerts_array[tProcessAlert_local].OFFLINE_TIME_DELAY,
                processAlerts_array[tProcessAlert_local].OFFLINE_DEADBAND
            }
        }
        RESPONSE_CODES
        {
            0,  SUCCESS,              no_command_specific_errors_temp;
            2,  DATA_ENTRY_ERROR,     invalidSelectionProcessAlertEtAl_string;
            5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
            6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
            7,  MODE_ERROR,           in_write_protect_mode_temp;
            9,  DATA_ENTRY_ERROR,     highAlertValueTooLarge_string;
            10, DATA_ENTRY_ERROR,     highAlertValueTooSmall_string;
            11, DATA_ENTRY_ERROR,     lowAlertValueTooLarge_string;
            12, DATA_ENTRY_ERROR,     lowAlertValueTooSmall_string;
            13, DATA_ENTRY_ERROR,     timeDelayDeadbandTooLarge_string;
            15, DATA_ENTRY_ERROR,     timeDelayDeadbandTooSmall_string;
            16, MODE_ERROR,           accessRestricted_string;
        }
    }

    COMMAND setProcessAlertLogToDefault_command
    {
        NUMBER 201;
        OPERATION COMMAND;
        TRANSACTION
        {
            REQUEST
            {
                tProcessAlert_local (INFO,INDEX)
            }
            REPLY
            {
                response_code,
                device_status,
                tProcessAlert_local (INFO,INDEX)
```

```
            }
        }
        RESPONSE_CODES
        {
            0,  SUCCESS,              no_command_specific_errors_temp;
            2,  DATA_ENTRY_ERROR,     invalidSelectionProcessAlert_string;
            5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
            6,  MISC_ERROR,         deviceSpecificCommandErrorNVFailure_string;
            7,  MODE_ERROR,           in_write_protect_mode_temp;
            16, MODE_ERROR,           accessRestricted_string;
        }
    }

    COMMAND readProcessAlertSetup_command
    {
        NUMBER 202;
        OPERATION READ;
        TRANSACTION 0
        {
            REQUEST
            {
                tProcessAlert_local (INFO,INDEX)
            }
            REPLY
            {
                response_code,
                device_status,
                tProcessAlert_local (INFO,INDEX),
                processAlerts_array[tProcessAlert_local].NAME,
                processAlerts_array[tProcessAlert_local].VALUE_UNITS,
                processAlerts_array[tProcessAlert_local].NOTIFICATION_MODE,
                processAlerts_array[tProcessAlert_local].DEVICE_VARIABLE,
                processAlerts_array[tProcessAlert_local].ACTIVATION_TRIGGER,
                processAlerts_array[tProcessAlert_local].HIGH_VALUE,
                processAlerts_array[tProcessAlert_local].LOW_VALUE,
                processAlerts_array[tProcessAlert_local].SUPPRESSION,
                processAlerts_array[tProcessAlert_local].TIME_DELAY,
                processAlerts_array[tProcessAlert_local].DEADBAND,

processAlerts_array[tProcessAlert_local].HIGH_ALERT_WITH_DEADBAND,
```

```
processAlerts_array[tProcessAlert_local].LOW_ALERT_WITH_DEADBAND
            }
        }
        TRANSACTION 1
        {
            REQUEST
            {
                tProcessAlert_local (INFO,INDEX)
            }
            REPLY
            {
                response_code,
                device_status,
                tProcessAlert_local (INFO,INDEX),
                processAlerts_array[tProcessAlert_local].OFFLINE_NAME,
                processAlerts_array[tProcessAlert_local].OFFLINE_VALUE_UNITS,

processAlerts_array[tProcessAlert_local].OFFLINE_NOTIFICATION_MODE,

processAlerts_array[tProcessAlert_local].OFFLINE_DEVICE_VARIABLE,

processAlerts_array[tProcessAlert_local].OFFLINE_ACTIVATION_TRIGGER,
                processAlerts_array[tProcessAlert_local].OFFLINE_HIGH_VALUE,
                processAlerts_array[tProcessAlert_local].OFFLINE_LOW_VALUE,
                processAlerts_array[tProcessAlert_local].OFFLINE_SUPPRESSION,
                processAlerts_array[tProcessAlert_local].OFFLINE_TIME_DELAY,
                processAlerts_array[tProcessAlert_local].OFFLINE_DEADBAND,

processAlerts_array[tProcessAlert_local].HIGH_ALERT_WITH_DEADBAND,

processAlerts_array[tProcessAlert_local].LOW_ALERT_WITH_DEADBAND
            }
        }
        RESPONSE_CODES
        {
            0,  SUCCESS,              no_command_specific_errors_temp;
            2,  DATA_ENTRY_ERROR,     invalidSelectionProcessAlert_string;
            5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
        }
    }
```

```
COMMAND readProcessAlertMonitoredValueAndLog_command
{
    NUMBER 203;
    OPERATION READ;
    TRANSACTION
    {
        REQUEST
        {
            tProcessAlert_local (INFO,INDEX)
        }
        REPLY
        {
            response_code,
            device_status,
            tProcessAlert_local (INFO,INDEX),
            processAlerts_array[tProcessAlert_local].VALUE_UNITS,
            processAlerts_array[tProcessAlert_local].MONITORED_VALUE,
            processAlerts_array[tProcessAlert_local].STATUS,
            processAlerts_array[tProcessAlert_local].MAX_VALUE_SEEN,
            processAlerts_array[tProcessAlert_local].MIN_VALUE_SEEN,
            processAlerts_array[tProcessAlert_local].TOTAL_TIME
        }
    }
    RESPONSE_CODES
    {
        0,  SUCCESS,              no_command_specific_errors_temp;
        2,  DATA_ENTRY_ERROR,     invalidSelectionProcessAlert_string;
        5,  DATA_ENTRY_ERROR,     too_few_data_bytes_recieved_temp;
    }
}
```