ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р МЭК 61784-3-12— 2016

Промышленные сети

ПРОФИЛИ

Часть 3-12

Функциональная безопасность полевых шин. Дополнительные спецификации для CPF 12

(IEC 61784-3-12:2010, IDT)

Издание официальное

Москва
Стандартинформ
2017

gost.e-ecolog.ru

Предисловие

1 ПОДГОТОВЛЕН Обществом с ограниченной ответственностью «Корпоративные электронные системы» (КЭЛС) на основе собственного перевода на русский язык англоязычной версии международного стандарта, указанного в пункте 4

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 58 «Функциональная безопасность»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии России от 30 ноября 2016 г. № 1882-ст

4 Настоящий стандарт идентичен международному стандарту МЭК 61784-3-12:2010 «Промышленные сети. Профили. Часть 3-12. Функциональная безопасность полевых шин. Дополнительные спецификации для CPF 12» (IEC 61784-3-12:2010, Industrial communication networks — Profiles — Part 3-12: Functional safety fieldbuses — Additional specifications for CPF 12, IDT).

При применении настоящего стандарта рекомендуется использовать вместо ссылочных международных стандартов соответствующие им национальные и межгосударственные стандарты, сведения о которых приведены в дополнительном приложении ДА

5 ВВЕДЕН ВПЕРВЫЕ

*Правила применения настоящего стандарта установлены в статье 26 Федерального закона от 29 июня 2015 г. № 162-ФЗ «О стандартизации в Российской Федерации». Информация об изменениях к настоящему стандарту публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок — в ежемесячном информационном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ближайшем выпуске ежемесячного информационного указателя «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (*gost.e-ecolog.ru

© Стандартинформ, 2017

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии

Содержание

1 Область применения ... 1
2 Нормативные ссылки ... 1
3 Термины, определения, обозначения и сокращения ... 2
3.1 Термины и определения ... 2
3.2 Обозначения и сокращения ... 6
3.3 Условные обозначения ... 6
4 Обзор FSCP 12/1 (CC-Link-Safety™) ... 7
5 Общие положения ... 8
5.1 Внешние документы, предоставляющие спецификации для профиля ... 8
5.2 Функциональные требования безопасности ... 8
5.3 Меры безопасности ... 9
5.4 Структура коммуникационного уровня безопасности ... 9
5.5 Связи с FAL (и DLL, PhL) ... 10
6 Услуги коммуникационного уровня безопасности ... 10
6.1 Соединение FSoE ... 10
6.2 Цикл FSoE ... 11
6.3 Услуги FSoE ... 11
7 Протокол коммуникационного уровня безопасности ... 12
7.1 Формат PDU безопасности ... 12
7.2 Процедура коммуникаций FSCP 12/1 ... 17
7.3 Реакция на ошибки коммуникаций ... 26
7.4 Таблица состояний для ведущего устройства FSoE ... 27
7.5 Таблица состояний для ведомого устройства FSoE ... 44
8 Управление коммуникационным уровнем безопасности ... 68
8.1 Обработка параметров FSCP 12/1 ... 68
8.2 Параметры коммуникаций FSoE ... 68
9 Системные требования ... 69
9.1 Индикаторы и коммутаторы ... 69
9.2 Руководство по установке ... 71
9.3 Время реакции функции безопасности ... 71
9.4 Длительность запросов на обслуживание ... 74
9.5 Ограничения на вычисление характеристик системы ... 74
9.6 Техническое обслуживание ... 76
9.7 Руководство по безопасности ... 76
10 Оценка ... 76
Приложение А (справочное) Дополнительная информация для профилей коммуникаций функциональной безопасности CPF 12 ... 77
Приложение В (справочное) Информация для оценки профилей коммуникаций функциональной безопасности CPF 12 ... 82
Приложение ДА (справочное) Сведения о соответствии ссылочных международных стандартов национальным стандартам ... 83
Библиография ... 84

0 Введение

0.1 Общие положения

Стандарт МЭК 61158, посвященный полевым шинам, вместе с сопутствующими ему стандартами МЭК 61784-1 и МЭК 61784-2 определяет набор протоколов передачи данных, которые позволяют осуществлять распределенное управление автоматизированными приложениями. В настоящее время технология полевых шин считается общепринятой и хорошо себя зарекомендовала. Именно поэтому появляются многочисленные расширения, направленные на еще не стандартизированные области, такие как приложения реального времени, связанные с безопасностью и защитой.

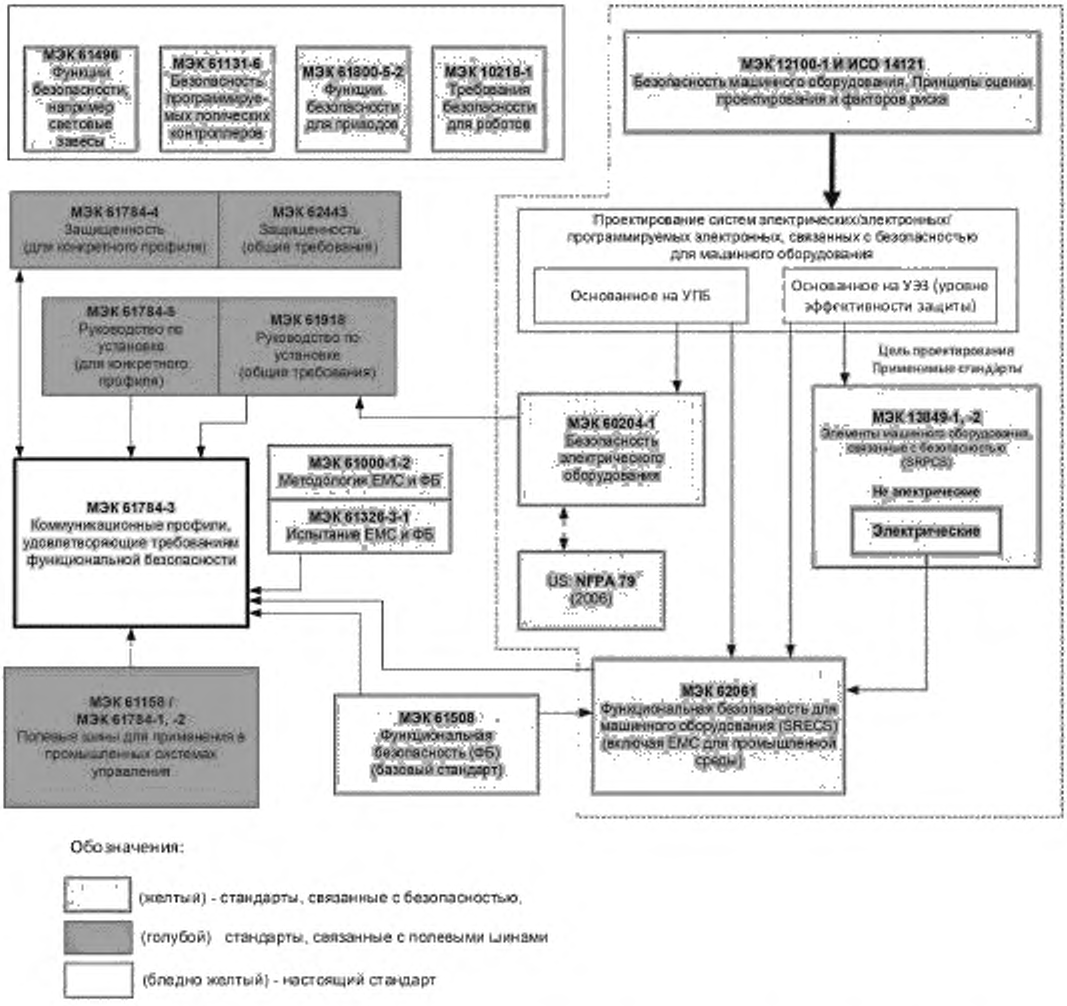
Настоящий стандарт рассматривает важные принципы функциональной безопасности коммуникаций на основе подхода, представленного в комплексе стандартов МЭК 61508, и определяет несколько коммуникационных уровней безопасности (профилей и соответствующих протоколов) на основе профилей передачи данных и уровней протоколов, описанных в МЭК 61784-1, МЭК 61784-2 и в комплексе стандартов МЭК 61158. Настоящий стандарт не рассматривает вопросы электробезопасности и искробезопасности.

На рисунке 1 представлена связь настоящего стандарта с соответствующими стандартами, посвященными функциональной безопасности и полевым шинам в среде машинного оборудования.

На рисунке 2 представлена связь настоящего стандарта с соответствующими стандартами, посвященным функциональной безопасности и полевым шинам в области промышленных процессов.

Коммуникационные уровни безопасности, реализованные в составе систем, связанных с безопасностью, в соответствии с МЭК 61508, обеспечивают необходимую достоверность при передаче сообщений (информации) между двумя и более участниками, использующими полевые шины в системе, связанной с безопасностью, или же достаточную уверенность в безопасном поведении при возникновении ошибок или отказов в полевой шине.

Коммуникационные уровни безопасности, определенные в настоящем стандарте, обеспечивают уверенность в том, что полевые шины могут использоваться в применениях, требующих обеспечение функциональной безопасности для конкретного уровня полноты функциональной безопасности (УПБ), для которого определен соответствующий ему профиль коммуникации, удовлетворяющий требованиям функциональной безопасности.



Примечание — Подразделы 6.7.6.4 (высокая степень сложности) и 6.7.8.1.6 (низкая степень сложности) МЭК 62061 устанавливают связь между PL (Категорией) и УПБ.

Рисунок 1 — Связь МЭК 61158-3 с другими стандартами (машинное оборудование)

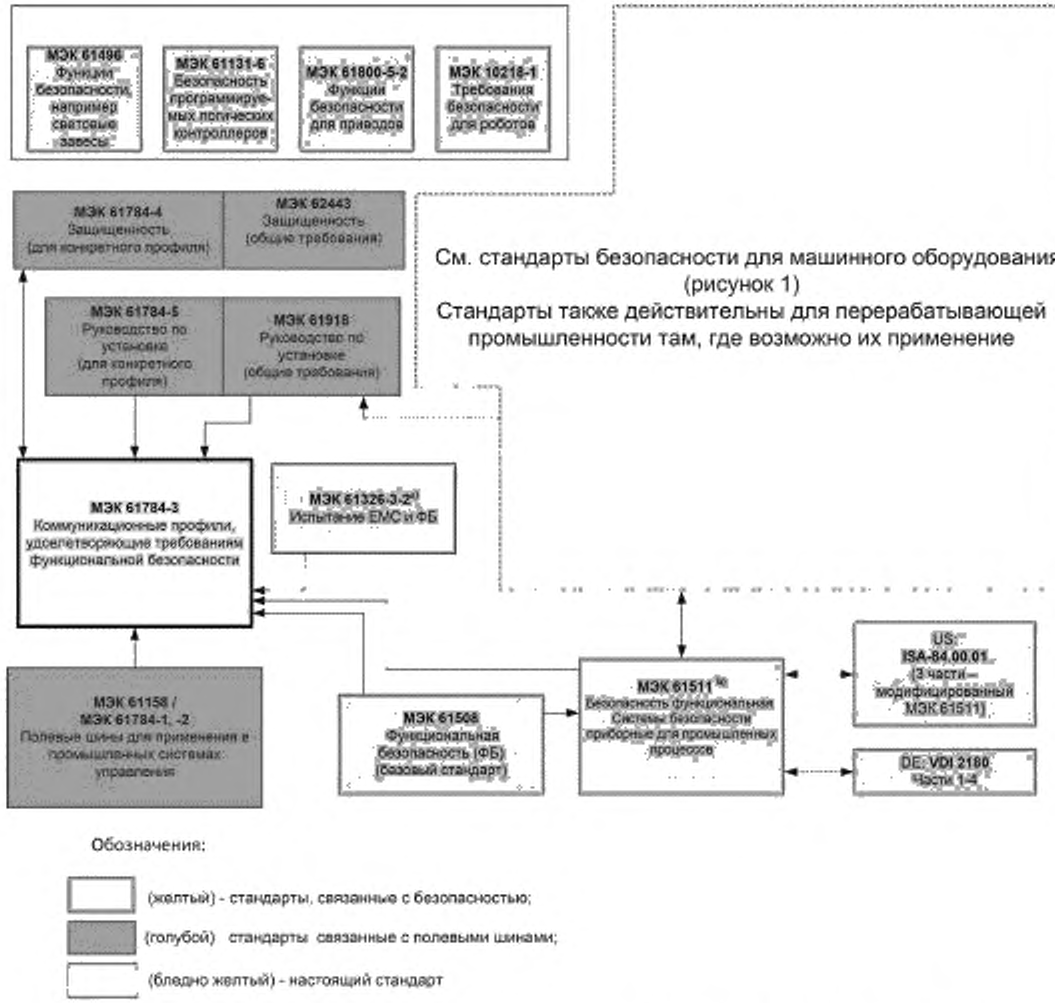


Рисунок 2 — Связь МЭК 61158-3 с другими стандартами (промышленные процессы)

Результирующий УПБ, заявляемый для системы, зависит от реализации выбранного профиля коммуникации, удовлетворяющего требованиям функциональной безопасности, внутри этой системы. Но реализации профиля коммуникации, удовлетворяющего требованиям функциональной безопасности, в стандартном устройстве недостаточно для того, чтобы устройство считалось устройством безопасности.

Настоящий стандарт описывает:

- основные принципы реализации требований комплекса стандартов МЭК 61508 для связанной с безопасностью передачи данных, включая возможные сбои при передаче данных, меры по устранению неисправностей и факторы, влияющие на полноту данных;

- индивидуальные описания профилей, удовлетворяющих требованиям функциональной безопасности, для нескольких семейств профилей передачи данных, представленных в МЭК 61784-1 и МЭК 61784-2;

- расширения уровня безопасности до служб передачи данных и разделов протоколов в стандартах комплекса МЭК 61158.

0.2 Декларация патента

Международный электротехнический комитет (МЭК) обращает внимание на то, что соблюдение требований настоящего стандарта может включать использование патентов, относящихся к профилям коммуникаций, соответствующих требованиям функциональной безопасности. Патенты для семейства 1 приведены ниже, где обозначение [xx] указывает на держателя патента:

DE 10 2004 044 764.0	[SI]	Datenübertragungsverfahren und Automatisierungssystem zum Einsatz eines solchen Datenübertragungsverfahrens
EP 05 733 921.0	[SI]	Sicherheitssteuerung

МЭК не занимается подтверждением обоснованности, подтверждением соответствия и областью применения прав данных патентов.

Правообладатели на данные патенты заверили МЭК, что они готовы рассмотреть использование лицензий на разумных и недискриминационных условиях и положениях с заявителями по всему миру. Такие заявления обладателей прав на данные патенты зарегистрированы в МЭК.

Информация доступна в:

[SI] Beckhoff Automation GmbH
Eiserstrasse 5, 33415 Verl
GERMANY

Обращаем ваше внимание на то, что некоторые элементы настоящего стандарта могут быть субъектом патентных прав, отличных от указанных ранее. МЭК не несет ответственности за идентификацию (частично или полностью) подобных патентных прав.

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Промышленные сети

ПРОФИЛИ

Часть 3-12

Функциональная безопасность полевых шин. Дополнительные спецификации для CPF 12

Industrial communication networks. Profiles. Part 3-12. Functional safety fieldbuses. Additional specifications for CPF 12

Дата введения — 2018—01—01

1 Область применения

Настоящий стандарт описывает коммуникационный уровень безопасности (услуги и протокол) на основе CPF 12, представленного в МЭК 61784-2 и МЭК 61158, Тип 12. Настоящий стандарт идентифицирует принципы коммуникаций, удовлетворяющих требованиям функциональной безопасности, определенных в МЭК 61784-3, что важно для этого коммуникационного уровня безопасности.

Примечание — Настоящий стандарт не затрагивает вопросы электробезопасности и искробезопасности. Электробезопасность связана с угрозами, такими как электрический шок. Искробезопасность связана с угрозами, относящимися к возможным взрывам в атмосфере.

Настоящий стандарт определяет механизмы передачи важных для безопасности сообщений между участниками распределенной сети, использующей технологию полевых шин, в соответствии с требованиями функциональной безопасности, представленными в комплексе МЭК 61508[1)]. Эти механизмы могут широко использоваться в промышленности, например в управлении процессом автоматизации производства и в машинном оборудовании.

Настоящий стандарт содержит руководства как для разработчиков, так и для оценщиков соответствующих приборов и систем.

Примечание — Результирующий УПБ, заявляемый для системы, зависит от реализации выбранного профиля коммуникации, удовлетворяющей требованиям функциональной безопасности, внутри этой системы. Но в соответствии с настоящим стандартом реализации выбранного профиля коммуникации, удовлетворяющей требованиям функциональной безопасности, в стандартном устройстве недостаточно для того, чтобы устройство считалось устройством безопасности.

2 Нормативные ссылки

Ссылки на следующие незаменимые для данного документа стандарты представлены ниже. Для датированных ссылок применяют только указанное издание ссылочного документа, для недатированных ссылок применяют последнее издание ссылочного документа (включая все его изменения).

IEC 60204-1, Safety of machinery — Electrical equipment of machines — Part 1: General requirements (Безопасность машинного оборудования. Электрическое оборудование машин. Часть 1. Общие требования)

IEC 61000-6-2, Electromagnetic compatibility (EMC) — Part 6-2: Generic standards — Immunity for industrial environments (Совместимость технических средств электромагнитная. Устойчивость к электромагнитным помехам технических средств, применяемых в промышленных зонах. Часть 6-2. Требования и методы испытаний)

[1)] Далее в настоящем стандарте используется «МЭК 61508» вместо «комплекс МЭК 61508».

Издание официальное

IEC 61131-2, Programmable controllers — Part 2: Equipment requirements and tests (Программируемые контроллеры. Часть 2. Требования к оборудованию и тестирование)

IEC 61158-2, Industrial communication networks — Fieldbus specifications — Part 2: Physical layer specification and service definition (Промышленные сети связи. Спецификации полевых шин. Часть 2: Спецификация физического уровня и определение сервиса)

IEC 61158-3-12, Industrial communication networks — Fieldbus specifications — Part 3-12: Data-link layer service definition — Type 12 elements (Промышленные сети связи. Спецификации полевых шин. Часть 3-12: Определение сервиса канального уровня. Элементы типа 12)

IEC 61158-3-12, Industrial communication networks — Fieldbus specifications — Part 3-12: Data-link layer protocol definition — Type 12 elements (Промышленные сети связи. Спецификации полевых шин. Часть 3-12: Определение протокола канального уровня. Элементы типа 12)

IEC 61158-5-12, Industrial communication networks — Fieldbus specifications — Part 5-12: Application layer service definition — Type 12 elements (Промышленные сети связи. Спецификации полевых шин. Часть 5-12: Определение сервиса прикладного уровня. Элементы типа 12)

IEC 61158-6-12, Industrial communication networks — Fieldbus specifications — Part 6-12: Application layer protocol specification — Type 12 elements (Промышленные сети связи. Спецификации полевых шин. Часть 6-12: Спецификация протокола прикладного уровня. Элементы типа 12)

IEC 61326-3-1, Electrical equipment for measurement, control and laboratory use — EMC requirements — Part 3-1: Immunity requirements for safety-related systems and for equipment intended to perform safety related functions (functional safety) — General industrial applications (Электрооборудование для измерений, управления и лабораторного применения. Часть 3-1. Требования защищенности для систем, связанных с безопасностью, и для оборудования, предназначенного для выполнения функций, связанных с безопасностью (функциональная безопасность). Общие промышленные приложения)

IEC 61326-3-2, Electrical equipment for measurement, control and laboratory use — EMC requirements — Part 3-2: Immunity requirements for safety-related systems and for equipment intended to perform safety related functions (functional safety) — Industrial applications with specified electromagnetic environment (Электрооборудование для измерений, управления и лабораторного применения. Часть 3-1. Требования защищенности для систем, связанных с безопасностью, и для оборудования, предназначенного для выполнения функций, связанных с безопасностью (функциональной безопасностью). Промышленные приложения с определенной электромагнитной средой)

IEC 61508 (all parts), Functional safety of electrical/electronic/programmable electronic safety-related systems (Функциональная безопасность систем электрических/электронных/программируемых электронных, связанных с безопасностью)

IEC 61784-2, Industrial communication networks — Profiles — Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC 8802-3 (Промышленные сети. Профили. Часть 2. Дополнительные профили полевых шин для сетей реального времени, основанные на ИСО/МЭК 8802-3)

IEC 61784-3, Industrial communication networks — Profiles — Part 3: Functional safety fieldbuses — General rules and profile definitions (Промышленные сети. Профили. Часть 3. Функциональная безопасность полевых шин. Общие правила и определения профиля)

IEC 61918, Industrial communication networks — Installation of communication networks in industrial premises (Промышленные сети. Установка сетей связи в промышленных помещениях)

3 Термины, определения, обозначения и сокращения

3.1 Термины и определения

В настоящем стандарте используются следующие термины и определения:

3.1.1 Общепринятые термины и определения

3.1.1.1 **готовность** (availability): Вероятность того, что в течение заданного промежутка времени в автоматизированной системе не наблюдается неисправных состояний в системе, приводящих к потере производительности.

3.1.1.2 **черный канал** (black channel): Канал связи, для которого отсутствуют доказательства того, что проектирование и подтверждение соответствия были выполнены в соответствии с МЭК 61508.

3.1.1.3 **канал связи** (communication channel): Логическое соединение между двумя оконечными точками в коммуникационной системе.

3.1.1.4 **коммуникационная система** (communication system): Система (устройство), состоящая из технических средств, программного обеспечения и среды распространения, которая обеспечивает передачу сообщений (прикладной уровень по ИСО/МЭК 7498) от одного приложения другому.

3.1.1.5 **соединение** (connection): Логическое связывание между двумя прикладными объектами в одном или в разных устройствах.

3.1.1.6 **циклический контроль избыточности** [Cyclic Redundancy Check (CRC)] Получаемые из блока данных (значений) избыточных данных, которые запоминаются и передаются вместе с этим блоком данных, для обнаружения искажения данных. Процедура (метод), использующаяся для вычисления избыточных данных.

Примечания

1 Термины «CRC код» и «CRC подпись» и обозначения, такие как «CRC 1» и «CRC 2», также могут применяться в данном стандарте в отношении избыточных данных.

2 См. также [32], [33].

3.1.1.7 **ошибка** (error) Расхождение между вычисленным, наблюдаемым или измеренным значением или условием и истинным, установленным или теоретически верным значением или условием.

[МЭК 61508-4:2010], [МЭК 61158]

Примечания

1 Ошибки могут возникнуть вследствие ошибок проектирования аппаратных средств / программного обеспечения и/или вследствие искажения данных, вызванного электромагнитными помехами и/или другими воздействиями.

2 Ошибки не обязательно являются причиной отказов или сбоев.

3.1.1.8 **отказ** (failure) Прекращение способности функционального блока выполнять необходимую функцию либо функционирование этого блока любым способом, отличным от требуемого.

Примечание — В МЭК 61508-4 приведено такое же определение, но дополнено примечаниями.

[МЭК 61508-4:2010, модифицирован], [ИСО/МЭК 2382-14.01.11, модифицирован]

Примечание — Причиной отказа может служить ошибка (например, проблема, связанная с проектированием программного обеспечения/аппаратных средств или с нарушением при передаче сообщений).

3.1.1.9 **сбой** (fault): Ненормальный режим, который может вызвать снижение или потерю способности функционального блока выполнять требуемую функцию.

Примечание — Международный электротехнический словарь (191-05-01) определяет «сбой» как состояние, характеризуемое неспособностью выполнить необходимую функцию, исключая неспособность, возникающую во время профилактических работ или других плановых мероприятий, либо в результате недостатка внешних ресурсов.

[МЭК 61508-4:2010, модифицирован], [ИСО/МЭК 2382-14.01.10, модифицирован]

3.1.1.10 **полевая шина** (fieldbus): Коммуникационная система, основанная на последовательной передаче данных и применяющаяся в промышленной автоматизации или приложениях управления процессами.

3.1.1.11 **система полевых шин** (fieldbus system): Система, использующая полевую шину с подключенными устройствами.

3.1.1.12 **кадр** (frame): Упрощенный синоним для DLPDU (Блок данных протокола канала передачи данных)

3.1.1.13 **последовательность проверки кадра** [frame check sequence (FCS)]: Дополнительные данные, полученные из блока данных DLPDU (кадра) с помощью хеш-функции, которые запоминаются и передаются вместе с этим блоком данных для обнаружения искажения данных.

Примечания

1 Значение FCS может быть получено, используя, например, CRC или другую хеш-функцию.

2 См. также [34], [35].

3.1.1.14 **хеш-функция** (hash function): (Математическая) функция, которая преобразует значения из (вероятно, очень) большого набора значений в (обычно) меньший диапазон значений.

Примечания

1 Хеш-функции могут применяться для обнаружения искажений данных.

2 Распространенные хеш-функции включают в себя контроль четности, вычисление контрольной суммы или CRC.

[МЭК/ТО 62210, модифицирован]

3.1.1.15 **опасность** (hazard): Состояние или набор условий в системе, которые вместе с другими, связанными с этими, условиями неизбежно приведут к причинению вреда человеку, имуществу или окружающей среде.

3.1.1.16 **ведущее устройство** (master): Активный объект коммуникации, способный инициировать и управлять во времени коммуникационной деятельностью других станций, которые могут быть как ведущими, так и ведомыми.

3.1.1.17 **сообщение** (message): Упорядоченные последовательности октет, предназначенные для передачи информации.

[ИСО/МЭК 2382-16.02.01, модифицирован].

3.1.1.18 **уровень эффективности защиты; УЭЗ** [performance level (PL)]: Дискретный уровень, применяющийся для определения способности связанных с безопасностью частей системы управления выполнять функцию безопасности в прогнозируемых условиях.

[ИСО 13849-1]

3.1.1.19 **защитное сверхнизкое напряжение** (protective extra-low-voltage, PELV): Электрическая цепь, в которой значение напряжения не может превышать среднеквадратичное значение переменного напряжения в 30 В, пиковое напряжение 42,4 В или постоянное напряжение 60 В при нормальных условиях и одиночном сбое за исключением короткого замыкания на землю в других цепях.

Примечание — Электрическая цепь PELV аналогична цепи SELV с защитным заземлением.

[МЭК 61131-2]

3.1.1.20 **избыточность** (redundancy): существование более одного средства выполнения необходимой функции или представления информации.

Примечание — В МЭК 61508-4 такое же определение, но дополнено примером и примечаниями.

[МЭК 61508-4:2010, модифицирован], [ИСО/МЭК 2382-14.01.12, модифицирован]

3.1.1.21 **надежность** (reliability). Вероятность того, что автоматизированная система может выполнять требующуюся функцию в заданных условиях на протяжении заданного промежутка времени (t_1, t_2).

Примечания

1 Принято считать, что автоматизированная система в состоянии выполнять данную требующуюся функцию в начале заданного промежутка времени.

2 Понятие «надежность» также используется для обозначения показателя надежности, измеряемого данной вероятностью.

3 На протяжении среднего времени между отказами (MTBF) или среднего времени до отказа (MTTF) вероятность того, что автоматизированная система выполнит требующуюся функцию, уменьшается.

4 Надежность отличается от готовности.

[МЭК 62059-11, модифицирован]

3.1.1.22 **риск** (risk) Сочетание вероятности события причинения вреда и тяжести этого вреда.

Примечание — Более подробно это понятие обсуждается в приложении А МЭК 61508-5:2010.

[МЭК 61508-4:2010], [ИСО/МЭК Руководство 51:1999, определение 3.2]

3.1.1.23 **коммуникационный уровень безопасности, КУБ** (safety communication layer, SCL): Уровень коммуникации, включающий все необходимые меры для обеспечения безопасной передачи информации в соответствии с требованиями МЭК 61508.

3.1.1.24 **данные безопасности** (safety data): Данные, передаваемые через безопасную сеть, используя протокол безопасности.

Примечание — Коммуникационный уровень безопасности не гарантирует безопасность самой информации, а только то, что она передается безопасно.

3.1.1.25 **устройство безопасности** (safety device): Устройство, спроектированное в соответствии с МЭК 61508 и реализующее профиль коммуникации, удовлетворяющий требованиям функциональной безопасности.

3.1.1.26 **безопасное сверхнизкое напряжение** (safety extra-low-voltage, SELV): Электрическая цепь, в которой значение напряжения не может превышать среднее квадратическое значение переменного напряжения в 30 В, пиковое напряжение 42,4 В или постоянное напряжение 60 В при нормальных условиях и одиночном сбое, включая короткое замыкание на землю в других цепях.

Примечание — Цепь SELV не подсоединена к защитному заземлению.

[МЭК 61131-2]

3.1.1.27 **функция безопасности** (safety function): Функция, реализуемая Э/Э/ПЭ (электрической, электронной, программируемой электронной) системой, связанной с безопасностью, или другими ме-

рами по снижению риска, предназначенная для достижения или поддержания безопасного состояния управляемого оборудования по отношению к конкретному опасному событию.

Примечание — В МЭК 61508-4 такое же определение, но дополнено примером и примечанием.

[МЭК 61508-4:2010, модифицирован]

3.1.1.28 **время реакции функции безопасности** (safety function response time): Наихудшее время между срабатыванием датчика системы безопасности, подключенного к полевой шине, и достижением соответствующего безопасного состояния с помощью необходимого исполнительного устройства этой системы безопасности при наличии ошибок или отказов в канале функции безопасности.

Примечание — Данная концепция введена в МЭК 61784-3:2010 и реализуется профилем коммуникации, удовлетворяющим требованиям функциональной безопасности, определенным в настоящем стандарте.

3.1.1.29 **уровень полноты безопасности, УПБ** (safety integrity level SIL): дискретный уровень (принимающий одно из четырех возможных значений), соответствующий диапазону значений полноты безопасности, при котором уровень полноты безопасности, равный 4, является наивысшим уровнем полноты безопасности, а уровень полноты безопасности, равный 1, соответствует наименьшей полноте безопасности.

Примечания

1 Целевые значения отказов (см. МЭК 61508-4:2010, п. 3.5.17) для четырех уровней полноты безопасности указаны в МЭК 61508-1:2010, таблицы 2 и 3.

2 Уровни полноты безопасности используют при определении требований полноты безопасности для функций безопасности, которые должны быть распределены по Э/Э/ПЭ системам, связанным с безопасностью.

3 Уровень полноты безопасности (УПБ) не является свойством системы, подсистемы, элемента или компонента. Правильная интерпретация фразы «УПБ системы, связанной с безопасностью, равен *n*» (где *n* = 1, 2, 3 или 4) означает: система потенциально способна к реализации функций безопасности с уровнем полноты безопасности до значения, равного *n*.

[МЭК 61508-4:2010]

3.1.1.30 **мера безопасности** (safety measure) <в данном стандарте> средство управления возможными ошибками коммуникаций, спроектированное и реализованное в соответствии с требованиями МЭК 61508.

Примечания

1 На практике, как правило, объединяют несколько мер безопасности для достижения требуемого уровня полноты безопасности.

2 Ошибки коммуникаций и связанные с ними меры безопасности подробно рассмотрены в МЭК 61784-3:2010, 5.3 и 5.4.

3.1.1.31 **приложение, связанное с безопасностью** (safety-related application): Программы, разработанные в соответствии с МЭК 61508 и удовлетворяющие требованиям УПБ приложения.

3.1.1.32 **система, связанная с безопасностью** (safety-related system): система, выполняющая функцию безопасности в соответствии с МЭК 61508.

3.1.1.33 **ведомое устройство** (slave): Пассивный объект коммуникации, способный принимать сообщения и отправлять их в ответ на другой объект коммуникации, который может быть ведомым или ведущим.

3.1.1.34 **ложное аварийное отключение** (spurious trip): Аварийное отключение, вызванное системой безопасности, без запроса от процесса.

3.1.2 CPF 12. Дополнительные термины и определения

3.1.2.1 **отказоустойчивые данные** (fail-safe data): Выражение для данных, которые, в случае инициализации или ошибки, принимают заранее определенное значение.

Примечание — В настоящем стандарте значение отказоустойчивых данных всегда должно равняться «0».

3.1.2.2 **соединение FSoE** (FSoE Connection): Уникальная связь между ведущим устройством FSoE и ведомым устройством FSoE.

3.1.2.3 **цикл FSoE** (FSoE Cycle): Коммуникационный цикл, включающий один PDU ведущего устройства и соответствующее PDU ведомого устройства.

3.1.2.4 **Безопасный ввод** (SafeInput): Данные процесса безопасности, передаваемые от ведомого устройства FSoE ведущему устройству FSoE.

3.1.2.5 **Безопасный вывод** (SafeOutput): Данные процесса безопасности, передаваемые от ведущего устройства FSoE ведомому устройству FSoE.

3.1.2.6 **PDU ведущего устройства безопасности** (Safety Master PDU): PDU безопасности, передаваемое от ведущего устройства FSoE ведомому устройству FSoE.

3.1.2.7 **PDU ведомого устройства безопасности** (Safety Slave PDU): PDU безопасности, передаваемое от ведомого устройства FSoE ведущему устройству FSoE.

3.2 Обозначения и сокращения

3.2.1 Общие обозначения и сокращения

Сокращение	Полное выражение	Источник
CP	Профиль коммуникаций	[МЭК 61784-1]
CPF	Семейство профилей коммуникации	[МЭК 61784-1]
CRC	Циклический контроль избыточности	—
DLL	Уровень канала данных	[ИСО/МЭК 7498-1]
DLPDU	Блок данных протокола канала передачи данных	—
ЭМС	Электромагнитная совместимость	—
УО	Управляемое оборудование	[МЭК 61508-4:2010]
Э/Э/ПЭ	Электрические/электронные/программируемые электронные	[МЭК 61508-4:2010]
FAL	Прикладной уровень полевой шины (Fieldbus Application Layer)	[МЭК 61158-5]
FCS	Последовательность проверки кадра	—
ФБ	Функциональная безопасность	—
FSCP	Профиль коммуникации, удовлетворяющий требованиям функциональной безопасности	—
MTBF	Среднее время между отказами	—
MTTF	Среднее время до отказа	—
PDU	Блок данных протокола	[ИСО/МЭК 7498-1]
PELV	Защитное сверхнизкое напряжение	—
PhL	Физический уровень	[ИСО/МЭК 7498-1]
PL	Уровень эффективности защиты	[ИСО 13849-1]
PLC	Программируемый логический контроллер	—
SCL	Коммуникационный уровень безопасности	—
SELV	Безопасное сверхнизкое напряжение	—
УПБ	Уровень полноты безопасности	[МЭК 61508-4:2010]

3.2.2 CPF 12: Дополнительные термины и определения

SIS — Инструментальная система безопасности (safety instrumented systems)

Сокращение	Полное выражение
ASIC	Специализированная интегральная схема
FSoE	Отказоустойчивость по CPF 12
ID	Идентификатор
UML	Унифицированный язык моделирования

3.3 Условные обозначения

Условные обозначения, используемые для описаний объектов, услуг и протоколов, рассмотрены в МЭК 61158-3-12, МЭК 61158-4-12, МЭК 61158-5-12 и МЭК 61118-6-12.

При необходимости для описания концпеций настоящий стандарт использует структурные схемы и UML диаграммы последовательности действий.

Состояния в диаграммах состояний представлены прямоугольниками, переходы состояний показаны в виде стрелок. Названия состояний и переходов диаграммы состояний соответствуют названиям в текстовом списке переходов состояний. Третья колонка содержит действие(я), которые должны быть выполнены. Последняя колонка содержит следующее состояние.

Таблица 1 — Элементы описания конечного автомата

Переход	Условие	Действие	Следующее состояние

Каждое состояние и его переход описаны в отдельном подразделе. Для каждого события, которое может произойти в состоянии, вводится дополнительный подраздел.

4 Обзор FSCP 12/1 (CC-Link Safety™)

Серия 12 профилей коммуникаций (общеизвестная как EtherCAT™[1)]) определяет профили коммуникаций, основанные на МЭК 61158-2,Тип 12, МЭК 61158-3-12, МЭК 61158-4-12, МЭК 61158-5-12 и МЭК 61158-6-12.

Базовые профили CP 12/1 и CP 12/2 определены в МЭК 61784-2. Коммуникационный профиль, удовлетворяющий требованиям функциональной безопасности, FSCP 12/1 (Safety-over-EtherCAT™[1)]) серии CPF 12 основан на базовых профилях CPF 12 из МЭК 61784-2, а также на спецификациях коммуникационного уровня безопасности, определенных в настоящем стандарте.

FSCP 12/1 описывает протокол для пересылки данных безопасности до УПБ 3 между устройствами FSCP 12/1. PDU безопасности пересылаются подчиненной полевой шиной, которая не включена в требования обеспечения безопасности, так как она может считаться черным каналом. PDU безопасности, которыми обмениваются два партнера по связи, воспринимаются подчиненной полевой шиной как данные процесса, которыми они циклически обмениваются.

FSCP 12/1 использует уникальную связь ведущего/ведомого между ведущим и ведомым устройствами FSoE, она называется соединение FSoE (рисунок 3). В соединении FSoE каждое устройство, как только получает новое сообщение от устройства-партнера, возвращает только свое собственное новое сообщение. Полный путь пересылки между ведомым устройством FSoE и ведущим устройством FSoE наблюдается отдельным сторожевым таймером, установленным на обоих устройствах в каждом цикле FSoE.

Ведущее устройство FSoE может обрабатывать более одного соединения FSoE для поддержки нескольких ведомых устройств FSoE.

[1)] EtherCAT™ и Safety-over-EtherCAT™ являются торговыми марками Beckhoff, Verl. Данная информация приведена для удобства использования данного международного стандарта и не означает, что МЭК поддерживает мнение обладателя торговой марки или его продукцию. Соответствие этому стандарту не требует использования наименований EtherCAT™ и Safety-over-EtherCAT™. Использование торговых марок EtherCAT™ и Safety-over-EtherCAT™ требует разрешения со стороны Beckhoff, Verl.

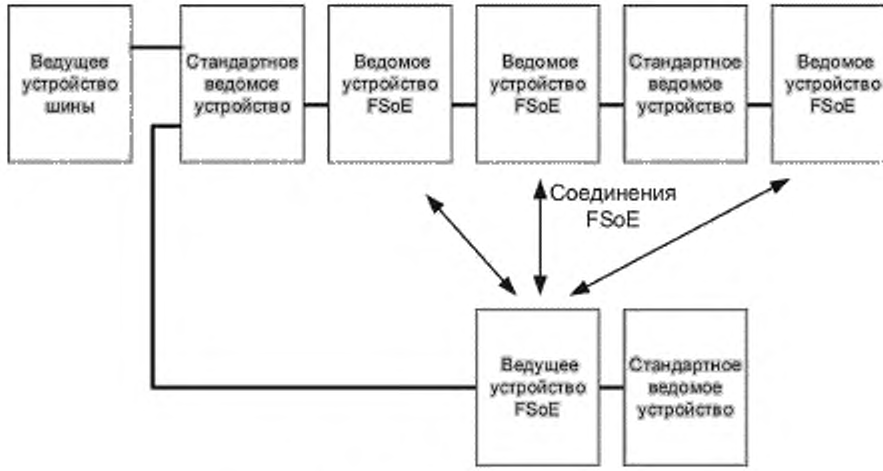


Рисунок 3 — Базовая система FSCP 12/1

Целостность передачи данных безопасности обеспечивается следующим образом:

- номер сеанса для обнаружения буферизации завершенной последовательности загрузки;
- порядковый номер для определения коммутации, повторения, внесения или потери целых сообщений;
- идентификация уникального соединения для безопасного обнаружения сообщения, ошибочно направленного по другому маршруту, с помощью уникальной связи адреса;
- сторожевой оперативный контроль для безопасного обнаружения недопустимых задержек на коммуникационном пути;
- проверка целостности данных циклическим избыточным кодом для обнаружения искажения сообщений от источника приемнику.

Смены состояний инициируются ведущим устройством FSoE и подтверждаются ведомым устройством FSoE. Машина состояний FSoE также предполагает обмен информацией и ее проверку для коммуникационной связи.

5 Общие положения

5.1 Внешние документы, предоставляющие спецификации для профиля

Нижеприведенный документ полезен для понимания конструкции протокола FSCP 12/1:

- GS-ET-26 [33].

5.2 Функциональные требования безопасности

Следующие требования применяются к разработке устройств, реализующих протокол FSCP 12/1. Те же требования были использован в разработке FSCP 12/1.

- Протокол FSCP 12/1 спроектирован для поддержки уровня полноты безопасности 3 (УПБ 3) (см. МЭК 61508).
- Реализации FSCP 12/1 должны соответствовать МЭК 61508.
- Базовые требования для разработки протокола FSCP 12/1 содержатся в МЭК 61784-3.
- Протокол FSCP 12/1 реализуется, используя метод черного канала; отсутствует какая-либо зависимость от стандартных коммуникационных профилей CPF 12. Оборудование для передачи данных, такое как контроллеры, ASIC схемы, каналы, соединительные устройства и т. д., должны избегать модификаций.
- Окружающие условия должны соответствовать общим требованиям автоматизации, в основном стандартам МЭК 61326-3-1, для испытаний запаса безопасности, если отсутствуют конкретные стандарты для самого изделия.

- Коммуникации безопасности и коммуникации, не связанные с безопасностью, должны быть независимы друг от друга. Тем не менее, устройства, не связанные с безопасностью, и устройства безопасности должны быть способны использовать один коммуникационный канал.

- Реализация протокола FSCP 12/1 должна быть ограничена оконечными устройствами коммуникаций (ведущим устройством FSoE и ведомым устройством FSoE).

- Между ведомым устройством FSoE и его ведущим устройством FSoE должна всегда присутствовать коммуникационная связь 1:1.

- Коммуникации безопасности не должны ограничивать минимальное время цикла коммуникационной системы.

5.3 Меры безопасности

Меры обеспечения безопасности, используемые в FSCP 12/1 для обнаружения коммуникационных ошибок, перечислены в таблице 2. Меры безопасности должны обрабатываться и контролироваться для каждого отдельного устройства безопасности.

Таблица 2 — Коммуникационные ошибки и механизмы их обнаружения

Ошибки коммуникаций	Меры обеспечения безопасности				
	Порядковый номер (см. 7 1.3.4)	Временное ожидание (см. 6.2)[a]	Аутентификация соединения (см. 7 2.2.4)[b]	Сообщение обратной связи (см. 7.2.1)	Обеспечение целостности данных (см. 7.1.3)
Искажение					x
Непреднамеренное повторение	x				x
Неверная последовательность	x				x
Потеря	x	x		x	x
Недопустимая задержка		x		x	x
Внесение	x				x
Подмена		x		x	x
Адресация			x		
Периодически повторяющие отказы памяти в коммутаторах	x				x

[a] В настоящем стандарте этот экземпляр именуется как «Сторожевой таймер FSoE».
[b] В настоящем стандарте этот экземпляр именуется как «ID соединения FSoE».

5.4 Структура коммуникационного уровня безопасности

Протокол FSCP 12/1 расположен поверх стандартного сетевого протокола. На рисунке 4 показано, как протокол связан с уровнем CPF 12. Уровень безопасности принимает данные безопасности, поступающие от приложения, связанного с безопасностью, и передает эти данные посредством протокола FSCP 12/1.

PDU безопасности, содержащий данные безопасности и требуемые меры обнаружения ошибок, входит в объекты данных процесса коммуникаций (PDO). Отображение в данных процесса коммуникационной системы и запуск коммуникационного конечного автомата не являются частью данного протокола безопасности.

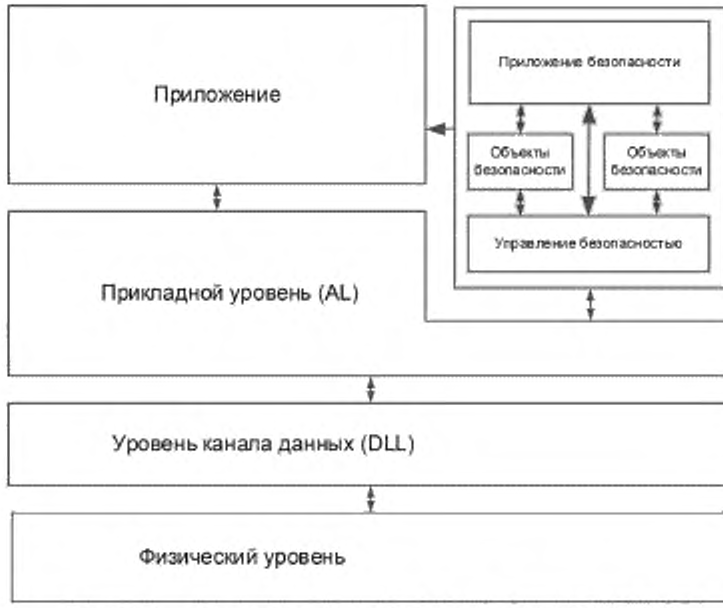


Рисунок 4 — Архитектура программного обеспечения FSCP 12/1

Вычисление вероятности возникновения остаточной ошибки для протокола FSCP 12/1 не связано с механизмами обнаружения ошибок коммуникационной системы. Это означает, что протокол можно также использовать для передачи в других коммуникационных системах. Может использоваться любой канал передачи, включая системы полевых шин, Ethernet или подобные сстемы передачи, волоконно-оптические кабели, медные провода или даже радиоканалы.

5.5 Связи с FAL (и DLL, PhL)

5.5.1 Общие положения

Коммуникационный уровень безопасности спроектирован для использования совместно с коммуникационными профилями CPF 12. Но он не ограничивается лишь этим коммуникационным профилем.

5.5.2 Типы данных

Профили, определенные в настоящем стандарте, поддерживают все типы данных CPF 12, заданные в МЭК 61158-5-12.

6 Услуги коммуникационного уровня безопасности

6.1 Соединение FSoE

Соединение между двумя коммуникационными партнерами по FSCP 12/1 (узлами с FSCP 12/1) именуется соединением FSoE. В соединении FSoE один коммуникационный партнер всегда является ведущим устройством FSoE, а другой ведомым устройством FSoE.

Ведущее устройство FSoE инициализирует соединение FSoE после включения питания или после сбоя коммуникации, в то время как ведомое устройство FSoE ограничивается ответами. Ведущее устройство FSoE устанавливает параметры коммуникаций, связанные с безопасностью, а также (не обязательно) параметры приложения, связанного с безопасностью, ведомого устройства FSoE.

Данные процесса безопасности, передаваемые от ведущего устройства FSoE ведомому устройству FSoE, называются безопасными выводами. Данные безопасности, передаваемые от ведомого устройства FSoE ведущему устройству FSoE, называются безопасными вводами.

PDU безопасности, передаваемый от ведущего устройства FSoE ведомому устройству FSoE, называется PDU безопасности ведущего устройства. PDU безопасности, передаваемый от ведомого устройства FSoE ведущему устройству FSoE, именуется PDU безопасности ведомого устройства.

6.2 Цикл FSoE

Ведущее устройство FSoE отправляет PDU безопасности ведущего устройства ведомому устройству безопасности FSoE и запускает сторожевой таймер FSoE.

После проверки целостности PDU безопасности ведомое устройство FSoE передает безопасные выводы приложению безопасности. Оно вычисляет PDU безопасности ведомого устройства с безопасными выводами, полученными от приложения безопасности и отправляет этот PDU ведущему устройству FSoE. Ведомое устройство FSoE также запускает свой сторожевой таймер FSoE. Это показано на рисунке 5.

После получения действительного PDU ведомого устройства цикл FSoE завершается.

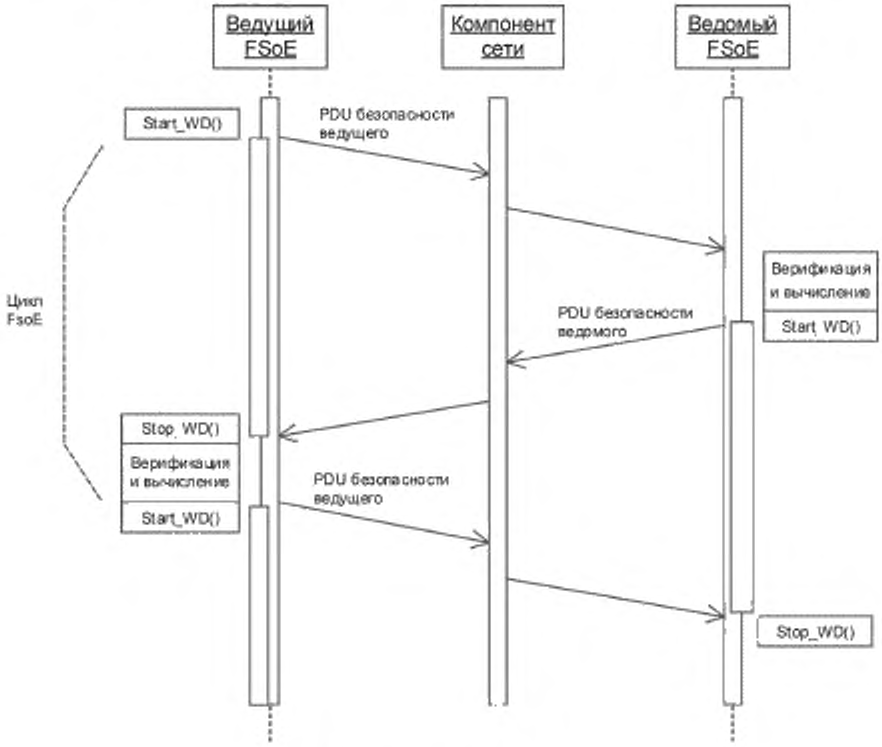


Рисунок 5 — Цикл FSoE

6.3 Услуги FSoE

Для каждого соединения FSoE ведущее устройство FSoE должно поддерживать обработчик ведущего устройства FSoE для управления связанным с ним ведомым устройством FSoE.

Для коммуникаций ведущего устройства FSoE с ведущим устройством FSoE ведущее устройство FSoE должно поддерживать один или несколько обработчиков ведомого устройства FSoE. На рисунке 6 показан возможный функционал FSoE для ведущих и ведомых устройств FSoE.

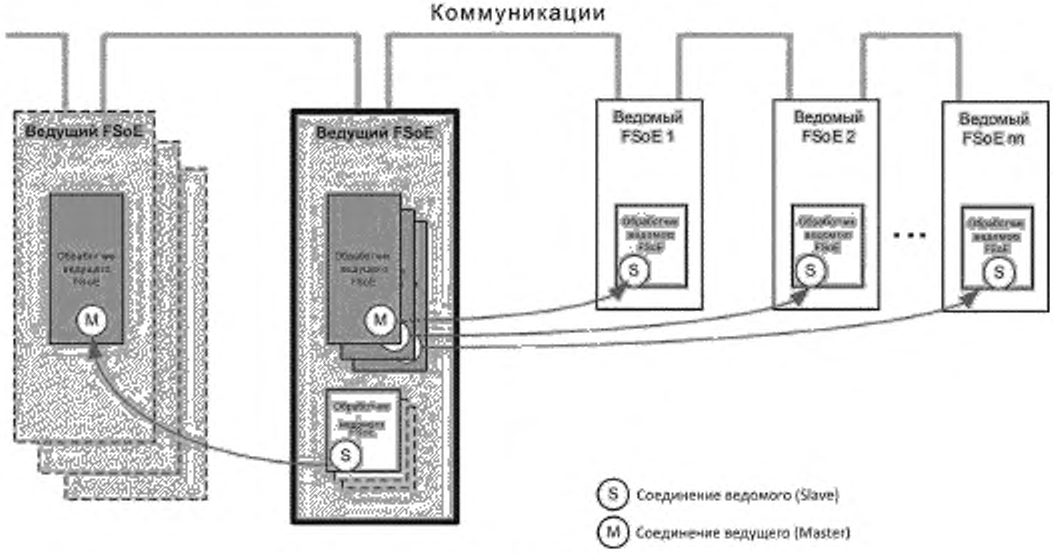


Рисунок 6 — Структура коммуникаций FSCP 12/1

7 Протокол коммуникационного уровня безопасности

7.1 Формат PDU безопасности

7.1.1 Структура PDU безопасности

На рисунке 7 показана структура одного PDU безопасности, встроенного в PDU Типа 12. В таблице 3 представлена общая структура PDU безопасности.

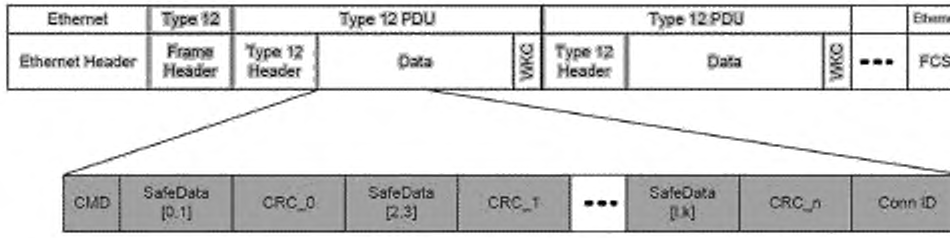


Рисунок 7 — PDU безопасности для CPF 12, встроенное в PDU Типа 12

PDU безопасности циклически передается через подчиненные полевые шины. Каждый узел FSCP 12/1 обнаруживает новый PDU безопасности, если хотя бы один бит в PDU безопасности был изменен.

PDU безопасности обладает переменной длиной, установленной в описании ведомого устройства FSoE.

Длина данных безопасности может составлять 1 октет или же четное число октет. Длина данных безопасности может отличаться в зависимости от направления (ввод или вывод).

Более короткая из двух длин данных безопасности в PDU ведущего устройства безопасности и PDU ведомого устройства безопасности определяет, как много данных безопасности используется во время фазы инициализации соединения FSoE с информацией о параметрах. Для более длинного из двух PDU блоков данных безопасности назначается значение ноль.

Таблица 3 — Общий PDU безопасности

Октет	Название	Описание
0	Command	Команда
1	SafeData[0]	данные безопасности, октет 0
2	SafeData[1]	данные безопасности, октет 1
3	CRC_0_Lo	младший октет (биты 0-7) 16-битовый CRC_0
4	CRC_0_Hi	старший октет (биты 8-15) 16-битовый CRC_0
5	SafeData[2]	данные безопасности, октет 2
6	SafeData[3]	данные безопасности, октет 3
7	CRC_1_Lo	младший октет (биты 0-7) 16-битовый CRC_1
8	CRC_1_Hi	старший октет (биты 8-15) 16-битовый CRC_1
…	…	
(*n*-1) × 2-1	SafeData[n-2]	данные безопасности, октет n-2
(*n*-1) × 2	SafeData[n-1]	данные безопасности, октет n-1
(*n*-1) × 2+1	CRC_(n-2)/2_Lo	младший октет (биты 0-7) 16-битовый CRC_(*n*-2)/2
(*n*-1) × 2+2	CRC_(n-2)/2_Hi	старший октет (биты 8-15) 16-битовый CRC_(*n*-2)/2
(*n*-1) × 2+3	Conn_Id_Lo	уникальный Id соединения, младший октет
(*n*-1) × 2+4	Conn_Id_Hi	уникальный Id соединения, старший октет

PDU безопасности может передавать *n* октетов данных безопасности. Два октета данных передаются с помощью 2-октетного CRC.

Короткий PDU безопасности состоит из 6 октетов, которые могут использоваться для передачи 1 октета данных безопасности, как это показано в таблице 4.

Таблица 4 — Короткий PDU безопасности

Октет	Название	Описание
0	Command	Команда
1	SafeData[0]	данные безопасности, октет 0
2	CRC_0_Lo	младший октет (биты 0-7) 16-битовый CRC_0
3	CRC_0_Hi	старший октет (биты 8-15) 16-битовый CRC_0
4	Conn_Id_Lo	уникальный Id соединения, младший октет
5	Conn_Id_Hi	уникальный Id соединения, старший октет

7.1.2 Команда PDU безопасности

Команда PDU безопасности определяет значение данных безопасности, основываясь на схеме, показанной в таблице 5.

Таблица 5 — Команда PDU безопасности

Команда	Описание
0x36	ProcessData (ДанныеПроцесса)
0x2A	Reset (Перезапуск)
0x4E	Session (Сеанс)
0x64	Connection (Соединение)
0x52	Parameter (Параметр)
0x08	FailSafeData (ОтказоустойчивыеДанные)

7.1.3 CRC блока PDU безопасности

7.1.3.1Вычисление CRC

Два октета данных безопасности передаются с помощью соответствующих двух октетов CRC.

Кроме передаваемых данных (command, data, ConnID), CRC_0 блока PDU безопасности также включает виртуальный порядковый номер, CRC_0 последнего полученного PDU безопасности и три дополнительных нулевых октета. Если передаются только данные безопасности одного октета, то SafeData[1] не учитывается в вычислении.

```
CRC_0 :=          f(received CRC_0, ConnID, Sequence_Number, command,
                  SafeData[0],SafeData[1], 0x000000)
```

В таблице 6 показана последовательность вычислений CRC_0.

Таблица 6 — Последовательность вычислений CRC_0

Шаг	Аргумент
1	полученный CRC_0 (бит 0-7)
2	полученный CRC_0 (бит 8-15)
3	ConnID (бит 0-7)
4	ConnID (бит 8-15)
5	Sequence_Number (бит 0-7)
6	Sequence_Number (бит 8-15)
7	Command
8	SafeData[0]
9	SafeData[1]
10	0
11	0
12	0

CRC_i (0 < i <= ((n-2)/2)) блока PDU безопасности также включает индекс CRC — i.

```
CRC_i :=          f(received CRC_0, ConnID, Sequence_Number, command, i,
                  SafeData[i × 2], SafeData[i × 2 + 1], 0)
```

В таблице 7 показана последовательность вычислений CRC_i.

Таблица 7 — Последовательность вычислений CRC_i (i>0)

Шаг	Аргумент
1	полученный CRC_0 (бит 0-7)
2	полученный CRC_0 (бит 8-15)
3	ConnID (бит 0-7)
4	ConnID (бит 8-15)
5	Sequence_Number (бит 0-7)
6	Sequence_Number (бит 8-15)
7	Command
8	Индекс i (bit 0-7)
9	Индекс i (bit 8-15)
10	SafeData[0]
11	SafeData[1]
12	0
13	0
14	0

7.1.3.2 Выбор полинома CRC

Полином 0x139B7 используется для вычисления сигнатур CRC и называется полиномом безопасности.

Для того чтобы разрешить передачу PDU безопасности по черному каналу, чьи характеристики передачи не связаны с безопасностью, при определении вероятности возникновения остаточной ошибки должна использоваться частота битовых сбоев, равная 10^{-2}. Вероятность возникновения остаточной ошибки не должна превышать 10^{-9}.

Безопасность обеспечивается за счет того, что ведущее устройство FSoE и ведомое устройство FSoE переходят в состояние сброса (т. е. безопасное состояние), как только обнаруживается ошибка.

Все параметры вычисления CRC за исключением данных безопасности обладают фиксированным ожидаемым значением, чтобы в вычислении вероятности остаточной ошибки учитывались только данные безопасности.

Математическое доказательство, демонстрирующее, что вероятность возникновения остаточной ошибки в случае полинома безопасности для 16-битовых данных безопасности и частоты битовых сбоев, равной 10^{-2}, не превышает 10^{-9}, включено в отдельный документ, где представлена полная количественная оценка.

7.1.3.3 Наследование CRC

Включение (наследование) CRC_0 последней полученной телеграммы в вычисление CRC гарантирует, что два последовательных PDU блока ведущего устройства безопасности или PDU ведомого устройства безопасности отличаются друг от друга, даже если другие данные не претерпели изменений.

Наследование CRC_0 также гарантирует безопасную и постоянную передачу данных, распространяемых несколькими PDU блоками Типа 12 по причине их длины.

CRC_0 полученного PDU безопасности включено в вычисление всех CRC_i для PDU безопасности, которое будет отправлено.

В таблице 8 показан пример для наследования CRC_0.

Таблица 8 — Пример наследования CRC_0

Цикл FSoE	Ведущее устройство FSoE		Ведомое устройство FSoE	
	старый CRC_0	новый CRC_0	старый CRC_0	новый CRC_0
j-1	CRC_0 (2 × j – 3)	CRC_0 (2 × j – 2)	CRC_0 (2 × j - 2)	CRC_0 (2 × j - 1)
j	CRC_0 (2 × j – 1)	CRC_0 (2 × j)	CRC_0 (2 × j)	CRC_0 (2 × j + 1)
j+1	CRC_0 (2 × j + 1)	CRC_0 (2 × j + 2)	CRC_0 (2 × j + 2)	CRC_0 (2 × j + 3)

В цикле FSoE j ведущее устройство FSoE получает PDU ведомого устройства безопасности с CRC_0 (2 × j – 1). Так как значение CRC_0 (2 × j – 2), которое было включено в вычисление CRC_0 (2 × j – 1), было вычислено ведущим устройством FSoE в FSoE цикле (j – 1), ведущее устройство FSoE может проверить CRC_0 (2 × j – 1) в PDU ведомого устройства безопасности.

В свою очередь, в FSoE цикле j ведомое устройство FSoE получает PDU ведущего устройства безопасности с CRC_0 (2 × j) и также способно проверять этот PDU, так как CRC_0 (2 × j - 1) вычисляется ведомым устройством FSoE в FSoE цикле (j – 1).

7.1.3.4 Порядковый номер

В таблице 8 CRC_0 (2 × j) может быть равен CRC_0 (2 × j – 2). В случае коротких PDU блоков безопасности это может привести к тому, что PDU ведущего устройства безопасности в FSoE цикле (j – 1) будет таким же, как и PDU ведушего устройства безопасности в FSoE цикле j, в результате чего ведомое устройство FSoE не признает PDU ведущего устройства безопасности как новый PDU в FSoE цикле j и сработает сторожевой таймер FSoE.

CRC коды PDU ведущего устройства безопасности, тем самым, включают виртуальный 16-битовый порядковый номер ведущего устройства, который ведущее устройство FSoE увеличивает с каждым новым PDU ведущего устройства безопасности. CRC код PDU ведомого устройства безопасности также включает виртуальный 16-битовый порядковый номер ведомого устройства, увеличиваемый ведомым устройством FSoE с каждым новым PDU ведомого устройства безопасности.

Если CRC_0 (2 × j) равен CRC_0 (2 × j – 2) несмотря на эти меры, то порядковый номер ведущего устройства увеличивается дальше до тех пор, пока CRC_0 (2 × j) не станет равным CRC_0 (2 × j – 2).

Такой алгоритм используется как для генерации ведущим устройством FSoE блока PDU ведущего устройства безопасности, так и для того, чтобы ведомое устройство FSoE могло проверить PDU ведущего устройства безопасности.

Если CRC_0 (2 × j + 1) равно CRC_0 (2 × j - 1), то порядковый номер ведомого устройства увеличивается дальше до тех пор, пока CRC_0 (2 × j + 1) не станет равен CRC_0 (2 × j - 1). Такой алгоритм используется как для генерации ведомым устройством FSoE блока PDU ведомого устройства безопасности, так и для того, чтобы ведущее устройство FSoE могло проверить PDU ведомого устройства безопасности.

Порядковый номер может принимать значения от 1 до 65 535. После 65 535 последовательность запускается снова, начиная с 1, т. е. 0 не учитывается.

7.1.3.5 Индекс CRC

Если передается более двух октетов данных безопасности и, тем самым, 2 или несколько кодов CRC (например CRC_0 и CRC_1), то мер, описанных выше, недостаточно для обнаружения всех возможностей для реверсирования в рамках PDU безопасности, см. пример в таблице 9.

Таблица 9 — Пример для 4 октетов данных безопасности с заменой октетов 1-4 на октеты 5-8.

Октет	Название	Описание
0	Command	Команда
1	SafeData[2]	данные безопасности, октет 2
2	SafeData[3]	данные безопасности, октет 3
3	CRC_1_Lo	младший октет (биты 0-7) 16-битовый CRC_1
4	CRC_1_Hi	старший октет (биты 8-15) 16-битовый CRC_1
5	SafeData[0]	данные безопасности, октет 0
6	SafeData[1]	данные безопасности, октет 1
7	CRC_0_Lo	младший октет (биты 0-7) 16-битовый CRC_0
8	CRC_0_Hi	старший октет (биты 8-15) 16-битовый CRC_0
9	Conn_Id_Lo	младший октет (биты 0-7) уникального Id соединения
10	Conn_Id_Hi	старший октет (биты 8-15) уникального Id соединения

Индекс i (двухоктетное значение), тем самым, также включается в соответствующий CRC_i. Это позволяет обнаруживать реверсирование октетов 1-4 и 5-8.

7.1.3.6 Дополнительные нулевые октеты

Вероятность возникновения остаточной ошибки вычисляется через соотношение обнаруженных и необнаруженных ошибок. Необнаруженные ошибки, по сути, являются ошибками, которые уже были обнаружены полиномом CRC для черного канала (стандартный полином), так как эти ошибки не очевидны на уровне безопасности, будучи отфильтрованными заранее. Наихудшее соотношение между обнаруженными ошибками (ошибками, не обнаруженными стандартным полиномом, но обнаруженные полиномом CRC уровня безопасности) и необнаруженными ошибками (ошибками, уже обнаруженными стандартным полиномом) возникает, если стандартный полином делится на полином безопасности без остатка.

В таком случае, для обеспечения надлежащей независимости двух полиномов друг от друга, в вычисление включаются три нулевых октета.

7.1.3.7 ID сеанса

Неисправное устройство, хранящее телеграммы (например, коммутатор), в особенности в случае полевых шин, передаваемых посредством Ethernet, может привести к тому, что правильная последовательность телеграмм вносится в неправильное время. Наследование CRC означает, что последовательность PDU безопасности всегда полагается на историю.

Передача произвольно генерируемого ID сеанса во время начальной настройки соединения FSoE гарантирует, что две последовательности PDU безопасности отличаются друг от друга после включения питания.

ID сеанса может принимать значения от 0 до 65 535.

7.2 Процедура коммуникаций FSCP 12/1

7.2.1 Цикл сообщения

Коммуникации FSCP 12/1 функционируют в рамках признанного цикла сообщения (FSoE цикл), т. е. ведущее устройство FSoE отправляет PDU ведущего устройства безопасности ведомому устройству FSoE и ожидает получить в ответ PDU безопасности. И только после этого генерируется следующий PDU ведущего устройства безопасности.

7.2.2 Состояния узлов FSCP 12/1

7.2.2.1 Общие положения

После установления соединения FSoE узлы FSCP 12/1 переходят в разные состояния перед тем, как данные безопасности становятся подтвержденными и состояние безопасности покидается.

На рисунке 8 показаны состояния узлов FSoE.

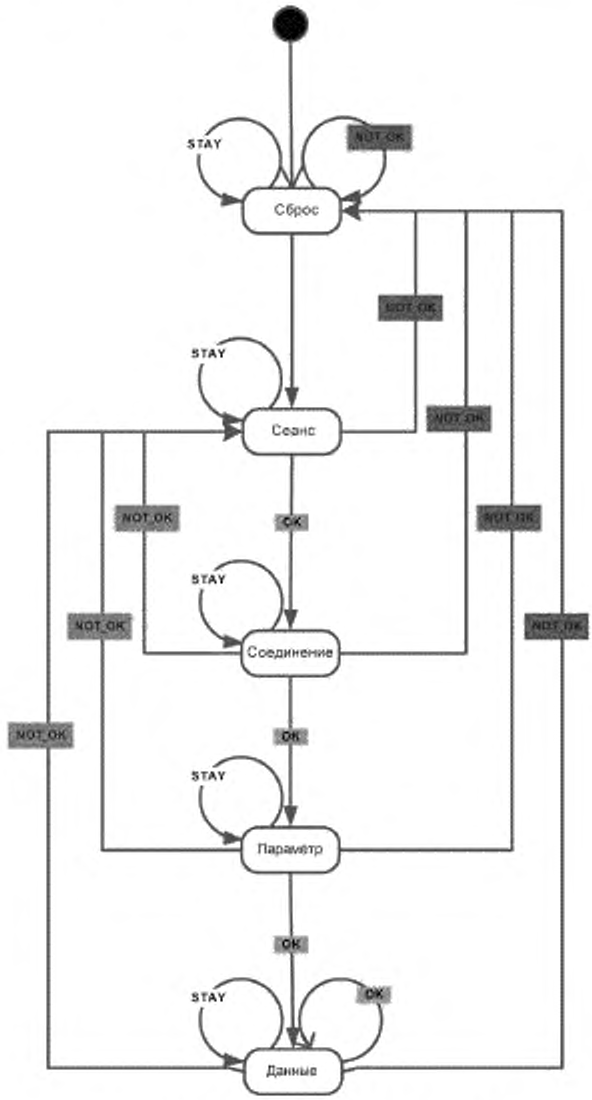


Рисунок 8 — Состояния узлов FSCP 12/1

После включения питания или ошибки коммуникаций ведущее и ведомое устройство FSoE находятся в состоянии сброса. Узлы FSoE также переключаются в reset-state (состояние сброса), если они обнаруживают ошибку в коммуникациях или локальном приложении. После выполнения команды FSoE Reset (сброс FSoE), поступившей от ведомого устройства FSoE, ведущее устройство FSoE переключается в состояние сеанса (переходы обозначены оранжевым цветом). После выполнения команды сброса, поступившей от ведущего устройства FSoE, ведомое устройство FSoE переключается в состояние сброса. Затем может быть принято состояние данных через состояния сеанса, соединения и параметров. Выход из состояния безопасного вывода может быть осуществлен только в состоянии данных.

7.2.2.2 Состояние сброса

Состояние сброса используется для повторной инициализации соединения FSoE после включения питания или возникновения ошибки коммуникаций FSoE. Ведущее устройство FSoE выходит из состояния сброса, когда оно отправляет PDU ведущего устройства безопасности с командой Session (сеанс) ведомому устройству FSoE. Ведомое устройство FSoE выходит из состояния сброса после того, как оно получает подтвержденный PDU ведущего устройства безопасности вместе с командной Session.

В состоянии сброса порядковый номер и CRC последней телеграммы, используемые в вычислении CRC, сбрасываются.

В таблице 10 показан пример PDU ведущего устройства безопасности для четырех октетов данных безопасности вместе с командой сброса.

Таблица 10 — PDU ведущего устройства безопасности для четырех октетов данных безопасности с command = Reset после сброса (сброса соединения, т. е. перезапуска) или ошибки

Октет	Название	Описание
0	Command	*Reset (сброс)*
1	SafeData[0]	код ошибки (бит 0-7), 0 для перезапуска
2	SafeData[1]	не используется (=0)
3	CRC_0_Lo	младший октет (биты 0-7) 16-битовый CRC_0
4	CRC_0_Hi	старший октет (биты 8-15) 16-битовый CRC_0
5	SafeData[2]	не используется (=0)
6	SafeData[3]	не используется (=0)
7	CRC_1_Lo	младший октет (биты 0-7) 16-битовый CRC_1
8	CRC_1_Hi	старший октет (биты 8-15) 16-битовый CRC_1
9	Conn_Id_Lo	не используется (=0)
10	Conn_Id_Hi	не используется (=0)

Ведомое устройство FSoE подтверждает команду Reset, устанавливая SafeData в значение 0. В таблице 11 показан пример PDU ведомого устройства безопасности для четырех октетов данных безопасности вместе с командой сброса.

Таблица 11 — PDU ведомого устройства безопасности для четырех октетов данных безопасности вместе с командой сброса

Октет	Название	Описание
0	Command	*Reset (сброс)*
1	SafeData[0]	0
2	SafeData[1]	0
3	CRC_0_Lo	младший октет (биты 0-7) 16-битовый CRC_0
4	CRC_0_Hi	старший октет (биты 8-15) 16-битовый CRC_0
5	SafeData[2]	не используется (=0)

Окончание таблицы 11

Октет	Название	Описание
6	SafeData[3]	не используется (=0)
7	CRC_1_Lo	младший октет (биты 0-7) 16-битовый CRC_1
8	CRC_1_Hi	старший октет (биты 8-15) 16-битовый CRC_1
9	Conn_Id_Lo	не используется (=0)
10	Conn_Id_Hi	не используется (=0)

Ведомое устройство FSoE также отправляет PDU безопасности с командой Reset во время перезапуска (сброс соединения) или в случае возникновения ошибки. Это показано в таблице 12, как пример для четырех октетов данных безопасности.

Таблица 12 — PDU ведомого устройства безопасности для четырех октетов данных безопасности с command = Reset после перезапуска (сброс соединения) или ошибки

Октет	Название	Описание
0	Command	*Reset (сброс)*
1	SafeData[0]	код ошибки (бит 0-7), 0 для сброса
2	SafeData[1]	не используется (=0)
3	CRC_0_Lo	младший октет (биты 0-7) 16-битовый CRC_0
4	CRC_0_Hi	старший октет (биты 8-15) 16-битовый CRC_0
5	SafeData[2]	не используется (=0)
6	SafeData[3]	не используется (=0)
7	CRC_1_Lo	младший октет (биты 0-7) 16-битовый CRC_1
8	CRC_1_Hi	старший октет (биты 8-15) 16-битовый CRC_1
9	Conn_Id_Lo	не используется (=0)
10	Conn_Id_Hi	не используется (=0)

Ведущее устройство FSoE подтверждает команду Reset, отправляя PDU ведущего устройства безопасности с командой Session.

7.2.2.3 Состояние сеанса

Во время перехода в состояние сеанса или при нахождении в нем, 16-битовый ID сеанса ведущего устройства передается от ведущего устройства FSoE ведомому устройству FSoE, которое в ответ возвращает свой собственный ID сеанса ведомого устройства.

Оба узла FSoE генерируют ID сеанса как произвольный номер, используемый для различия множественных последовательностей PDU безопасности в случае нескольких перезапусков соединения FSoE.

В таблице 13 показан пример PDU ведущего устройства безопасности для четырех октетов данных безопасности с командой Session.

Таблица 13 — PDU ведущего устройства безопасности для четырех октетов данных безопасности с command = Session

Октет	Название	Описание
0	Command	*Session (сеанс)*
1	SafeData[0]	*Id* Сеанса ведущего устройства, октет 0
2	SafeData[1]	*Id* Сеанса ведущего устройства, октет 1

Окончание таблицы 13

Октет	Название	Описание
3	CRC_0_Lo	младший октет (биты 0-7) 16-битовый CRC_0
4	CRC_0_Hi	старший октет (биты 8-15) 16-битовый CRC_0
5	SafeData[2]	не используется (=0)
6	SafeData[3]	не используется (=0)
7	CRC_1_Lo	младший октет (биты 0-7) 16-битовый CRC_1
8	CRC_1_Hi	старший октет (биты 8-15) 16-битовый CRC_1
9	Conn_Id_Lo	не используется (=0)
10	Conn_Id_Hi	не используется (=0)

Ведомое устройство FSoE подтверждает команду Session, отправляя назад ID сеанса ведомого устройства. В таблице 14 показан пример PDU ведомого устройства безопасности для четырех октетов SafeData (БезопасныеДанные) с командой Session.

Таблица 14 — PDU ведомого устройства безопасности для четырех октетов SafeData (БезопасныеДанные) с command = Session

Октет	Название	Описание
0	Command	*Session (сеанс)*
1	SafeData[0]	*Id* Сеанса ведомого устройства, октет 0
2	SafeData[1]	*Id* Сеанса ведомого устройства, октет 1
3	CRC_0_Lo	младший октет (биты 0-7) 16-битовой CRC_0
4	CRC_0_Hi	старший октет (биты 8-15) 16-битовой CRC_0
5	SafeData[2]	не используется (=0)
6	SafeData[3]	не используется (=0)
7	CRC_1_Lo	младший октет (биты 0-7) 16-битовой CRC_1
8	CRC_1_Hi	старший октет (биты 8-15) 16-битовой CRC_1
9	Conn_Id_Lo	не используется (=0)
10	Conn_Id_Hi	не используется (=0)

Если PDU безопасности содержит хотя бы 2 октета данных безопасности, ID сеанса может быть передан с одним FSoE циклом. Если, с другой стороны, PDU безопасности содержит только 1 октет данных безопасности, то ID сеанса должен быть передан с двумя FSoE циклами.

Значение ID сеанса не имеет никакого значения для безопасности, т. е. коммутатор в узле FSoE, получающем PDU безопасности, не нуждается в анализе проблем безопасности. Таким образом, ID соединения для команды Session устанавливается в значение 0.

Ведущее устройство FSoE выходит из состояния сеанса после того, как оно передало полный ID сеанса и получило связанные с ним подтверждения от ведомого устройства FSoE, отправив PDU безопасности с командой Connection (соединение) ведомому устройству FSoE. Ведомое устройство FSoE выходит из состояния сеанса после того, как оно получает PDU безопасности с командой Connection от ведущего устройства FSoE.

И ведущее, и ведомое устройства FSoE также выйдут из состояния сеанса, если они обнаружат ошибку коммуникаций FSoE.

В ведущем устройстве FSoE после получения команды RESET и порядковый номер, и CRC последней телеграммы, используемые в вычислении CRC, сбрасываются.

7.2.2.4 Состояние соединения

В состоянии соединения 16-битовый ID соединения передается от ведущего устройства FSoE ведомому устройству FSoE. ID соединения должен быть уникальным и сгенерированным конфигуратором ведущего устройства FSoE. Если в коммуникационной системе присутствует несколько

ведущих устройств FSoE, то пользователь должен убедиться, что используемые идентификаторы ID соединения уникальны.

В дополнение к 16-битовому ID соединения передается также уникальный адрес ведомого устройства FSoE. В таблице 15 показано содержание данных безопасности, передаваемых в состоянии соединения.

Таблица 15 — Данные безопасности, передаваемые в состоянии соединения

Октет данных безопасности	Описание
0	младший октет (биты 0-7) ID соединения
1	старший октет (биты 8-15) ID соединения
2	младший октет (биты 0-7) Адреса ведомого устройства FSoE
3	старший октет (биты 8-15) Адреса ведомого устройства FSoE

В зависимости от длины данных безопасности, требуется до четырех FSoE цикла. Для PDU безопасности с четырьмя октетами данных безопасности требуется только один FSoE цикл, что показано в таблицах 16 и 17.

Таблица 16 — PDU ведущего устройства безопасности для четырех октетов данных безопасности в состоянии соединения

Октет	Название	Описание
0	Command	*Connection (соединение)*
1	SafeData[0]	Id Соединения, младший октет
2	SafeData[1]	Id Соединения, старший октет
3	CRC_0_Lo	младший октет (биты 0-7) 16-битовой CRC_0
4	CRC_0_Hi	старший октет (биты 8-15) 16-битовой CRC_0
5	SafeData[2]	Адреса ведомого устройства FSoE, младший октет
6	SafeData[3]	Адреса ведомого устройства FSoE, старший октет
7	CRC_1_Lo	младший октет (биты 0-7) 16-битовой CRC_1
8	CRC_1_Hi	старший октет (биты 8-15) 16-битовой CRC_1
9	Conn_Id_Lo	Id соединения, младший октет
10	Conn_Id_Hi	Id соединения, старший октет

Ведомое устройство FSoE подтверждает команду Connection отправкой назад данных безопасности.

Таблица 17 — PDU ведомого устройства безопасности для четырех октетов данных безопасности в состоянии соединения.

Октет	Название	Описание
0	Command	Connection (соединение)
1	SafeData[0]	Id соединения, младший октет
2	SafeData[1]	Id соединения, старший октет
3	CRC_0_Lo	младший октет (биты 0-7) 16-битовой CRC_0
4	CRC_0_Hi	старший октет (биты 8-15) 16-битовой CRC_0
5	SafeData[2]	Адреса ведомого устройства FSoE, младший октет
6	SafeData[3]	Адреса ведомого устройства FSoE, старший октет
7	CRC_1_Lo	младший октет (биты 0-7) 16-битовой CRC_1
8	CRC_1_Hi	старший октет (биты 8-15) 16-битовой CRC_1
9	Conn_Id_Lo	Id соединения, младший октет
10	Conn_Id_Hi	Id соединения, старший октет

Адрес ведомого устройства FSoE должен быть уникальным в рамках коммуникационной системы. Он может быть установлен на соответствующем ведомом устройстве FSoE. Передавая адрес ведомого устройства FSoE вместе с ID соединения, ведомое устройство FSoE может проверять, было ли оно в действительности адресовано, для обнаружения недействительной адресации. Так как ID соединения также является уникальным в рамках коммуникационной системы, ID соединения всегда отправляет в последующих PDU блоках безопасности для того, чтобы и ведущее, и ведомое устройство FSoE могло обнаружить, адресуются ли им телеграммы. Тем самым, уникальный ID соединения включает 65 535 соединений FSoE для их реализации в коммуникационной системе (ID соединения = 0 не допускается).

7.2.2.5 Параметрическое состояние

В параметрическом состоянии осуществляется передача параметров коммуникаций, связанных с безопасностью, и приложений, связанных с безопасностью и зависящих от устройства. Приложения могут иметь произвольную длину. Наследование CRC гарантирует безопасность и постоянство передачи параметров.

В таблице 18 показано содержание данных безопасности, передаваемых в параметрическом состоянии.

Таблица 18 — Данные безопасности, передаваемые в параметрическом состоянии

Октет данных безопасности	Описание
0	младший октет (биты 0-7) длина параметров коммуникации в октетах (=2)
1	старший октет (биты 8-15) длина параметров коммуникации в октетах (=0)
2	младший октет (биты 0-7) сторожевого таймера FSoE (в мс)
3	старший октет (биты 8-15) сторожевого таймера FSoE (в мс)
4	младший октет (биты 0-7) длина параметров приложения в октетах
5	старший октет (биты 8-15) длина параметров приложения в октетах
6	1-й октет параметра приложения, связанного с безопасностью
…	
n + 5	*n*-й октет параметра приложения, связанного с безопасностью

Число циклов FSoE в параметрическом состоянии зависит от длины параметров приложения, связанного с безопасностью, и длины данных безопасности в PDU безопасности. Если не все октеты данных безопасности требуются в последнем FSoE цикле, то они должны передаваться как 0.

Для PDU безопасности с четырьмя октетами данных безопасности и двумя октетами параметров приложения, связанных с безопасностью, требуется два FSoE цикла; это показано в таблицах 19—22.

Таблица 19 — Первый PDU ведущего устройства безопасности для четырех октетов данных безопасности в параметрическом состоянии

Октет	Название	Описание
0	Command	*Parameter (*параметр*)*
1	SafeData[0]	младший октет (биты 0-7) длина параметров коммуникации в октетах (=2)
2	SafeData[1]	старший октет (биты 8-15) длина параметров коммуникации в октетах (=0)
3	CRC_0_Lo	младший октет (биты 0-7) 16-битовой CRC_0
4	CRC_0_Hi	старший октет (биты 8-15) 16-битовой CRC_0
5	SafeData[2]	младший октет (биты 0-7) сторожевого таймера FSoE (в мс)
6	SafeData[3]	старший октет (биты 8-15) сторожевого таймера FSoE (в мс)
7	CRC_1_Lo	младший октет (биты 0-7) 16-битовой CRC_1
8	CRC_1_Hi	старший октет (биты 8-15) 16-битовой CRC_1
9	Conn_Id_Lo	Id соединения, младший октет
10	Conn_Id_Hi	Id соединения, старший октет

Ведомое устройство FSoE подтверждает корректную команду Parameter возвращением данных безопасности.

Таблица 20 — Первый PDU ведомого устройства безопасности для четырех октетов данных безопасности в параметрическом состоянии

Октет	Название	Описание
0	Command	*Parameter (*параметр*)*
1	SafeData[0]	младший октет (биты 0-7) длина параметров коммуникации в октетах (=2)
2	SafeData[1]	старший октет (биты 8-15) длина параметров коммуникации в октетах (=0)
3	CRC_0_Lo	младший октет (биты 0-7) 16-битовой CRC_0
4	CRC_0_Hi	старший октет (биты 8-15) 16-битовой CRC_0
5	SafeData[2]	младший октет (биты 0-7) сторожевого таймера FSoE (в мс)
6	SafeData[3]	старший октет (биты 8-15) сторожевого таймера FSoE (в мс)
7	CRC_1_Lo	младший октет (биты 0-7) 16-битовой CRC_1
8	CRC_1_Hi	старший октет (биты 8-15) 16-битовой CRC_1
9	Conn_Id_Lo	Id соединения, младший октет
10	Conn_Id_Hi	Id соединения, старший октет

Ведущее устройство FSoE отправляет второй PDU ведущего устройства безопасности после того, как он корректно принял первый PDU ведомого устройства безопасности.

Таблица 21 — Второй PDU ведущего устройства безопасности для четырех октетов данных безопасности в параметрическом состоянии

Октет	Название	Описание
0	Command	*Parameter (*параметр*)*
1	SafeData[0]	младший октет (биты 0-7) длина параметров приложения в октетах (=2)
2	SafeData[1]	старший октет (биты 8-15) длина параметров приложения в октетах (=0)
3	CRC_0_Lo	младший октет (биты 0-7) 16-битовой CRC_0
4	CRC_0_Hi	старший октет (биты 8-15) 16-битовой CRC_0
5	SafeData[2]	1-й октет параметра приложения, связанного с безопасностью
6	SafeData[3]	2-й октет параметра приложения, связанного с безопасностью
7	CRC_1_Lo	младший октет (биты 0-7) 16-битовой CRC_1
8	CRC_1_Hi	старший октет (биты 8-15) 16-битовой CRC_1
9	Conn_Id_Lo	Id соединения, младший октет
10	Conn_Id_Hi	Id соединения, старший октет

Ведомое устройство FSoE подтверждает корректную команду Parameter возвращением данных безопасности.

Таблица 22 — PDU ведомого устройства безопасности для четырех октетов данных безопасности в параметрическом состоянии

Октет	Название	Описание
0	Command	*Parameter (*параметр*)*
1	SafeData[0]	младший октет (биты 0-7) длина параметров приложения в октетах (=2)
2	SafeData[1]	старший октет (биты 8-15) длина параметров приложения в октетах (=0)
3	CRC_0_Lo	младший октет (биты 0-7) 16-битовой CRC_0

Окончание таблицы 22

Октет	Название	Описание
4	CRC_0_Hi	старший октет (биты 8-15) 16-битовой CRC_0
5	SafeData[2]	1-й октет параметра приложения, связанного с безопасностью
6	SafeData[3]	2-й октет параметра приложения, связанного с безопасностью
7	CRC_1_Lo	младший октет (биты 0-7) 16-битовой CRC_1
8	CRC_1_Hi	старший октет (биты 8-15) 16-битовой CRC_1
9	Conn_Id_Lo	Id соединения, младший октет
10	Conn_Id_Hi	Id соединения, старший октет

Параметры сторожевого таймера FSoE и приложения, связанного с безопасностью, конфигурируются посредством конфигуратора безопасности ведущего устройства FSoE.

7.2.2.6 Состояние данных

7.2.2.6.1 Действительные (подтвержденные) данные

В то время как в предыдущих состояниях число FSoE циклов было фиксированным, в состоянии данных FSoE циклы передаются до тех пор, пока не возникнет коммуникационная ошибка или пока узел FSoE не будет локально остановлен. Ведущее устройство FSoE отправляет безопасные выводы ведомому устройству FSoE.

В таблице 23 показан пример PDU ведущего устройства для четырех октетов SafeOutputs с командой ProcessData (данные процесса).

Таблица 23 — PDU ведущего устройства безопасности для четырех октетов ProcessData в состоянии данных.

Октет	Название	Описание
0	Command	*ProcessData (*данные процесса*)*
1	SafeData[0]	1-й октет SafeOutputs
2	SafeData[1]	2-й октет SafeOutputs
3	CRC_0_Lo	младший октет (биты 0-7) 16-битовой CRC_0
4	CRC_0_Hi	старший октет (биты 8-15) 16-битовой CRC_0
5	SafeData[2]	3-й октет SafeOutputs
6	SafeData[3]	4-й октет SafeOutputs
7	CRC_1_Lo	младший октет (биты 0-7) 16-битовой CRC_1
8	CRC_1_Hi	старший октет (биты 8-15) 16-битовой CRC_1
9	Conn_Id_Lo	Id соединения, младший октет
10	Conn_Id_Hi	Id соединения, старший октет

Ведомое устройство FSoE подтверждает PDU ведущего устройства безопасности и отправляет SafeInputs ведущему устройству FSoE.

В таблице 24 показан пример PDU ведомого устройства безопасности для четырех октетов SafeInputs с командой ProcessData.

Таблица 24 — PDU ведомого устройства безопасности для четырех октетов ProcessData в состоянии данных.

Октет	Название	Описание
0	Command	*ProcessData (*данные процесса*)*
1	SafeData[0]	1-й октет SafeInputs
2	SafeData[1]	2-й октет SafeInputs
3	CRC_0_Lo	младший октет (биты 0-7) 16-битовой CRC_0

Окончание таблицы 24

Октет	Название	Описание
4	CRC_0_Hi	старший октет (биты 8-15) 16-битовой CRC_0
5	SafeData[2]	3-й октет SafeInputs
6	SafeData[3]	4-й октет SafeInputs
7	CRC_1_Lo	младший октет (биты 0-7) 16-битовой CRC_1
8	CRC_1_Hi	старший октет (биты 8-15) 16-битовой CRC_1
9	Conn_Id_Lo	Id соединения, младший октет
10	Conn_Id_Hi	Id соединения, старший октет

7.2.2.6.2 Команда FailSafeData

Если ведущее устройство FSoE локально обнаруживает, что безопасные выводы (SafeOutputs) не действительны или требуется перевести их в безопасное состояние, то оно отправляет команду FailSafeData.

В таблице 25 показан пример PDU ведущего устройства безопасности для четырех октетов FailsafeData с командой FailsafeData.

Таблица 25 — PDU ведущего устройства безопасности для четырех октетов отказоустойчивых данных в состоянии данных

Октет	Название	Описание
0	Command	*FailSafeData*
1	SafeData[0]	Отказоустойчивые данные = 0
2	SafeData[1]	Отказоустойчивые данные = 0
3	CRC_0_Lo	младший октет (биты 0-7) 16-битовой CRC_0
4	CRC_0_Hi	старший октет (биты 8-15) 16-битовой CRC_0
5	SafeData[2]	Отказоустойчивые данные = 0
6	SafeData[3]	Отказоустойчивые данные = 0
7	CRC_1_Lo	младший октет (биты 0-7) 16-битовой CRC_1
8	CRC_1_Hi	старший октет (биты 8-15) 16-битовой CRC_1
9	Conn_Id_Lo	Id соединения, младший октет
10	Conn_Id_Hi	Id соединения, старший октет

Если ведомое устройство FSoE локально обнаруживает, что безопасные вводы (SafeInputs) не действительны или требуется перевести их в безопасное состояние, то оно отправляет команду FailSafeData.

В таблице 26 показан пример PDU ведомого устройства безопасности для четырех октетов FailsafeData с командой FailsafeData.

Таблица 26 — PDU ведомого устройства безопасности для четырех октетов отказоустойчивых данных в состоянии данных

Октет	Название	Описание
0	Command	*FailSafeData*
1	SafeData[0]	Отказоустойчивые данные = 0
2	SafeData[1]	Отказоустойчивые данные = 0
3	CRC_0_Lo	младший октет (биты 0-7) 16-битовой CRC_0

Окончание таблицы 26

Октет	Название	Описание
4	CRC_0_Hi	старший октет (биты 8-15) 16-битовой CRC_0
5	SafeData[2]	Отказоустойчивые данные = 0
6	SafeData[3]	Отказоустойчивые данные = 0
7	CRC_1_Lo	младший октет (биты 0-7) 16-битовой CRC_1
8	CRC_1_Hi	старший октет (биты 8-15) 16-битовой CRC_1
9	Conn_Id_Lo	Id соединения, младший октет
10	Conn_Id_Hi	Id соединения, старший октет

Передача ProcessData или FailSafeData не зависит от команды полученного PDU безопасности. Она зависит только от локальных обстоятельств.

7.3 Реакция на ошибки коммуникаций

Узел FSoE может обнаруживать ошибки, перечисленные в таблице 27.

Таблица 27 — Коммуникационная ошибка FSoE

Ошибка	Описание
Неожиданная команда	Полученная команда не допускается в данном состоянии
Неизвестная команда	Полученная команда не определена
ID неправильного соединения	ID соединения не соответствует ID соединения, переданному в состоянии соединения
Ошибка CRC	Хотя бы один из полученных CRC_i не соответствует вычисленному CRC_i
Сторожевой таймер истек	За время сторожевого таймера не было получено ни одного действительного PDU безопасности
Недействительный адрес ведомого устройства FSoE	Адрес ведомого устройства FSoE, переданный в состоянии соединения, не соответствует локальному адресу, установленному на ведомом устройстве FSoE
Недействительные SafeData	Данные безопасности, возвращенные ведомым устройством FSoE в состояниях сеанса, соединения и параметров, не соответствуют ожидаемым значениям
Неисправные SafePara	SafePara, отправленные ведомому устройству FSoE в параметрическом состоянии, не действительны
Недопустимая длина параметров коммуникаций	Неправильная длина параметра коммуникаций
Недопустимый коммуникационный параметр	Неправильное содержание параметра коммуникаций
Недопустимая длина параметра приложения	Неправильная длина параметра приложения
Недопустимый параметр приложения	Неправильное содержание параметра приложения

Узел FSoE обнаруживает коммуникационную ошибку, отправляется команда Reset, а также связанный с ней код ошибки в SafeData[0] для диагностических целей. Ведущее устройство FSoE затем переключается в состоянии сеанса, ведомое устройство FSoE в состоянии сброса. Коды ошибок коммуникаций FSoE перечислены в таблице 28.

Таблица 28 — Коды ошибок коммуникаций FSoE

Октет	Описание
0	Локальный сброс или подтверждение команды сброса

Окончание таблицы 28

Октет	Описание
1	Неожиданная команда (INVALID_CMD)
2	Неизвестная команда (UNKNOWN_CMD)
3	Недействительное соединение (INVALID_CONNID)
4	Ошибка CRC (INVALID_CRC)
5	Сторожевой таймер истек (WD_EXPIRED)
6	Недействительный адрес ведомого устройства FSoE (INVALID_ADDRESS)
7	Недействительные данные безопасности (INVALID_DATA)
8	Недопустимая длина параметра коммуникаций (INVALID_COMMPARALEN)
9	Недействительные данные параметра коммуникаций (INVALID_COMPARA)
10	Недопустимая длина параметра приложения (INVALID_USERPARALEN)
11	Недействительные данные параметра приложения (INVALID_USERPARA)
0x80-0xFF	Недействительные SafePara (зависит от устройства)

7.4 Таблица состояний для ведущего устройства FSoE

7.4.1 Машина состояний ведущего устройства FSoE

7.4.1.1 Обзор

В зависимости коммуникационной процедуры ведущее устройство FSoE может находиться в состояниях, перечисленных в таблице 29.

Таблица 29 — Состояния ведущего устройства FSoE

Состояние	Описание
Сброс	Соединение FSoE сброшено (выводы в безопасном состоянии)
Сеанс	Передается ID сеанса (выводы в безопасном состоянии)
Соединение	Передается ID соединения (выводы в безопасном состоянии)
Параметры	Передаются параметры (выводы в безопасном состоянии)
Данные	Передаются данные процесса или отказоустойчивые данные (выводы активны, только если получена команда *ProcessData*)

Диаграмма состояний для ведущего устройства FSoE показана на рисунке 9.

В следующих секциях проводится анализ событий, которые могут произойти на ведущем устройстве FSoE для каждого состояния.

Каждое событие рассматривается в условиях различных действий или проистекающих состояний.

7.4.1.2 События

Событие может включать в себя различные параметры, на которые приводятся ссылки в таблицах состояний. В таблице 30 приведен список используемых событий.

Таблица 30 — События в таблице состояний ведущего устройства FSoE

Событие	Описание
Получен кадр	Был получен PDU безопасности, т. е. хотя бы один бит в PDU безопасности был изменен. Параметры: Frame — полученный PDU безопасности; Frame.Command — команда полученного PDU безопасности; Frame.Crc0 — CRC_0 полученного PDU безопасности; Frame.ConnId — ID соединения полученного PDU безопасности; Frame.SafeData — данные безопасности полученного PDU безопасности

Окончание таблицы 30

Событие	Описание
Истек сторожевой таймер	Истек сторожевой таймер FSoE, т. е. за время сторожевого таймера не было получено никаких PDU безопасности. Параметры: нет
Сброс Соединения	Запрос посредством локального интерфейса на сброс соединения FSoE. Данное событие должно возникать при включении питания для запуска коммуникаций с ведомым устройством FSoE. Параметры: нет
Команда Set Data	Запрос посредством локального интерфейса на переключение SafeOutputs в состояние безопасности или на выход из состояния безопасности. Параметры: DataCmd — *FailSafeData или ProcessData*

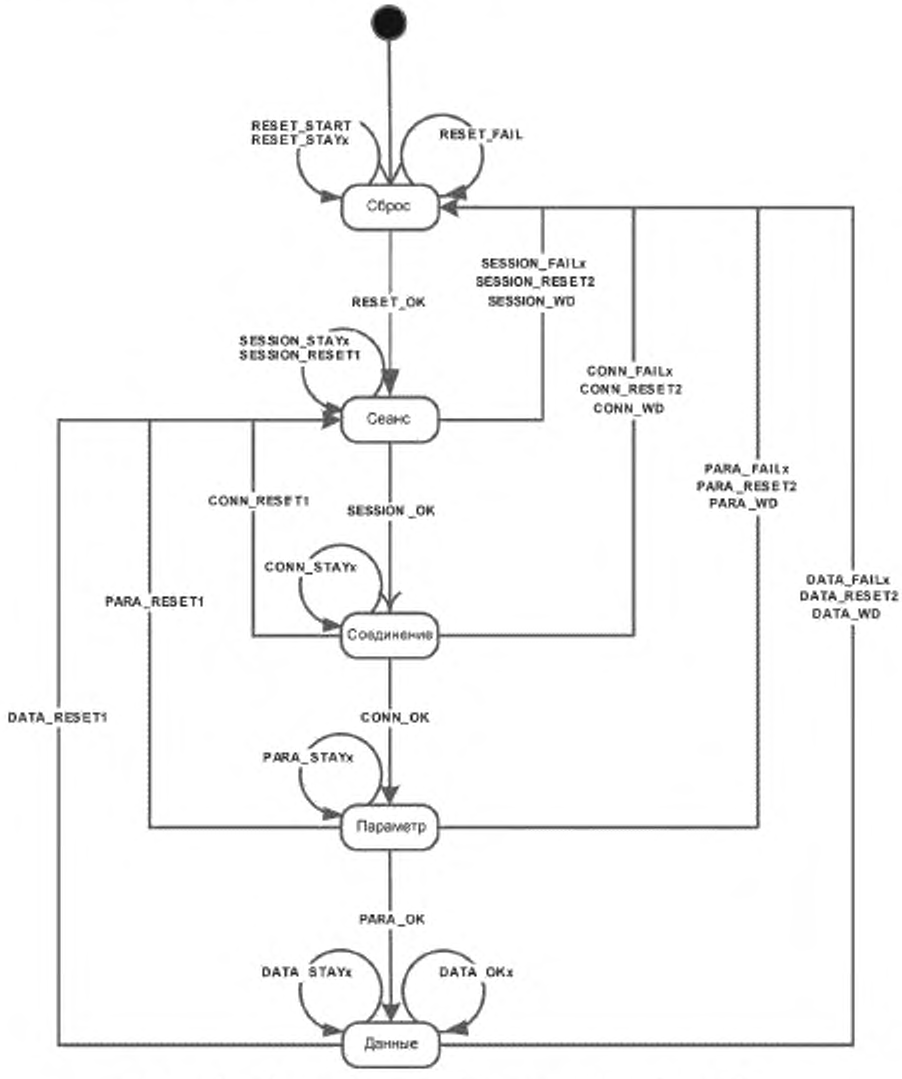


Рисунок 9 — Диаграмма состояний для ведущего устройства FSoE

7.4.1.3 Действия

В зависимости от разных условий выполняются определенные действия, если происходит событие. В таблицах состояний действия показаны в виде вызовов функций или присваиваний переменных.

В таблице 31 перечислены функции, используемые в таблице состояний ведущего устройства FSoE.

Таблица 31 — Функции в таблице состояний ведущего устройства FSoE

Функция	Описание
SendFrame(cmd, safeData, lastCrc, connId, seqNo, oldCrc, bNew)	Отправлен кадр ведущего устройства FSoE. Параметры: cmd — команда кадра; SafeData — ссылка на данные безопасности, отправленные с кадром; lastCrc — CRC_0 последнего PDU ведомого устройства безопасности, включенный в вычисление CRC для кадра; connId — ID соединения, которое необходимо ввести в кадр и включить в вычисление CRC; seqNo — Указатель на Порядковый номер ведущего устройства, включенный в вычисление CRC для кадров. Возвращается увеличенный (возможно неоднократно) seqNo; oldCrc: указатель на CRC_0 последнего отправленного PDU ведущего устройства безопасности. Возвращается вычисленный CRC_0.; bNew: если bNew = TRUE и oldCrc равен вычисленному crc, то вычисление CRC повторяется с приращенным seqNo до тех пор, пока вычисленный crc не станет равен oldCrc (процедура соответствует 7.1.3.4)

В таблице 32 перечислены переменные, используемые в таблице состояний ведущего устройства FSoE.

Таблица 32 — Переменные, используемые в таблице состояний ведущего устройства FSoE.

Состояние	Описание
LastCrc	CRC_0 последнего отправленного PDU ведущего устройства безопасности (инициализируется значением 0 при включении питания)
OldMasterCrc	CRC_0 последнего отправленного PDU ведущего устройства безопасности (инициализируется значением 0 при включении питания)
OldSlaveCrc	CRC_0 последнего полученного PDU ведомого устройства безопасности (инициализируется значением 0 при включении питания)
MasterSeqNo	Порядковый номер ведущего устройства для использования в CRC для следующего PDU ведущего устройства безопасности (инициализируется значением 0 при включении питания)
SlaveSeqNo	Ожидаемый порядковый номер ведомого устройства для использования в CRC следующего PDU ведомого устройства безопасности (инициализируется значением 0 при включении питания)
SessionId	Произвольно генерируемый ID сеанса (инициализируется значением 0 при включении питания)
DataCommand	Указывает на то, какая из команд *ProcessData* или *FailSafeData* отправлена в состоянии данных. Инициализируется с помощью *FailSafeData* при включении питания
BytesToBeSent	Если несколько PDU блоков безопасности должно быть отправлено в состоянии сеанса, соединения или параметров, то эта переменная указывает на то, сколько еще октетов должно быть отправлено (инициализируется значением 0 при включении питания)
ConnData	ConnData состоит из ID соединения и адреса ведомого устройства FSoE. Инициализируется конфигуратором безопасности при включении питания в соответствии с конфигурацией ConnData.ConnId: ConnectionId соединения FSoE

Окончание таблицы 32

Состояние	Описание
SafePara	SafePara состоит из параметров коммуникаций безопасности и приложения безопасности. Инициализируется конфигуратором безопасности при включении питания в соответствии с конфигурацией SafePara.Watchdog: сторожевой таймер FSoE
SafeParaSize	Указывает на длину SafePara. Инициализируется конфигуратором безопасности при включении питания в соответствии с данными конфигурации
SafeOutputs	Содержит значения процесса выводов безопасности, отправленное ведомому устройству FSoE. Инициализируется с помощью FS_VALUE (Fail-safe Data = 0) при включении питания
SafeInputs	Содержит значения процесса вводов безопасности, полученных ведомым устройством FSoE. Инициализируется с помощью FS_VALUE (Fail-safe Data = 0) при включении питания
CommFaultReason	Указывает на код ошибки в случае события возникновения коммуникационной ошибки
SecondSessionFrameSent	Если два блока PDU безопасности должны быть отправлены в состоянии Сеанса, то эта переменная указывает на то, был ли уже отправлен второй PDU. Данная переменная устанавливается в значение FALSE макрокомандой CREATE_SESSION_ID

7.4.1.4 Макрокоманды

Определенные функциональные возможности объединяются в макрокоманды для того, чтобы сохранять прозрачность таблиц состояний.

В таблице 33 перечислены макрокоманды из таблицы состояний ведущего устройства FSoE.

Таблица 33 — Макрокоманды в таблице состояний ведущего устройства FSoE

Функция	Описание
IS_CRC_CORRECT(frame, lastCrc, seqNo, oldCrc, bNew)	Эта макрокоманда проверяет корректны ли CRC коды полученного PDU ведомого устройства безопасности. Параметры: Frame — принятый кадр; lastCrc — CRC_0 последнего отправленного PDU ведущего устройства безопасности, включенное в вычисления CRC для принятого PDU; seqNo — порядковый номер ведомого устройства включается в вычисления CRC для принятого кадра. Возвращается приращенный (возможно неоднократно) seqNo; oldCrc: указатель на CRC_0 последнего принятого PDU ведомого устройства безопасности. Возвращается CRC_0 полученной телеграммы; bNew: если bNew = TRUE и oldCrc равняется вычисленному crc, то вычисление CRC повторяется с приращенным seqNo до тех пор, пока вычисленный crc не перестанет быть равным oldCrc (процедура выполняется согласно 7.1.3.4)
UPDATE_BYTES_TO_BE_SENT (bytesSent)	Данная макрокоманда проверят, сколько еще октетов в состояниях сеанса, соединения и параметров необходимо отправить перед изменением состояния. Параметры: bytesSent — число октетов, ждущих отправления
IS_SAFEDATA_CORRECT (frame, expectedData, bytesSent)	Данная макрокоманда проверяет, совпадает ли SafeData полученного PDU ведомого устройства безопасности с ожидаемыми данными. Параметры: Frame — принятый кадр; expectedData — ссылка на ожидаемые данные; bytesSent — число отправленных октетов

Окончание таблицы 33

Функция	Описание
START_WD (watchdog)	Данная макрокоманда перезапускает сторожевой таймер и запускает контролирующий таймер. Параметры: Watchdog — время контроля (мониторинга) в мс
CREATE_SESSION_ID	Данная макрокоманда генерирует произвольный ID сеанса. Переменная SecondSessionFrameSent (отправленный кадр второго сеанса) сбрасывается в значение FALSE
ADR	Данная макрокоманда генерирует ссылку (указатель) на переменную

7.4.2 Состояние сброса

7.4.2.1 Событие принятия кадра

Переход	Условие	Действие	Следующее состояние
RESET_OK	Frame.Command = Reset	SessionId := CREATE_SESSION_ID(); SendFrame(Session, ADR(SessionId), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); LastCrc = SendFrame.Crc0; BytesToBeSent := UPDATE_BYTES_TO_BE_SENT(2); START_WD(SafePara.Watchdog);	Session (Сеанс)
RESET_STAY1	Frame.Command <> Reset	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := FailSafeData; CommFaultReason := 0; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1;	Reset (Сброс)

7.4.2.2 Событие истекшего сторожевого таймера

Переход	Условие	Действие	Следующее состояние
RESET_WD	Истек сторожевой таймер	SessionId := CREATE_SESSION_ID(); SendFrame(Session, ADR(SessionId), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); LastCrc = SendFrame.Crc0; BytesToBeSent := UPDATE_BYTES_TO_BE_SENT(2); START_WD(SafePara.Watchdog);	Session (Сеанс)

7.4.2.3 Событие сброса соединения

Переход	Условие	Действие	Следующее состояние
RESET_START	Истек сторожевой таймер	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := FailSafeData; CommFaultReason := 0; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara.Watchdog);	Reset (Сброс)

7.4.2.4 Событие Команды Set Data

Переход	Условие	Действие	Следующее состояние
RESET_STAY2		DataCommand := DataCmd;	Reset (Сброс)

7.4.3 Состояние сеанса

7.4.3.1 Событие получения кадра

Переход	Условие	Действие	Следующее состояние
SESSION_OK	Frame.Command = *Session* AND BytesToBeSent = 0 AND IS_CRC_CORRECT(Frame, LastCrc, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE) = TRUE	LastCrc := Frame.Crc0; SendFrame(Connection, ADR(ConnData), LastCrc, ConnData.ConnId, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE); LastCrc := SendFrame.Crc0; BytesToBeSent := UPDATE_BYTES_TO_BE_SENT(4); START_WD(SafePara.Watchdog);	Connection (Соединение)
SESSION_FAIL1	Frame.Command = *Session* AND IS_CRC_CORRECT(Frame, LastCrc, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE) = FALSE AND SecondSessionFrameSent = TRUE	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CRC; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara.Watchdog);	Reset (Сброс)

Переход	Условие	Действие	Следующее состояние
SESSION_STAY2	Frame.Command = *Session* AND IS_CRC_CORRECT(Frame, LastCrc, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE) = FALSE AND SecondSessionFrameSent = FALSE	START_WD(SafePara.Watchdog);	Session (Сеанс)
SESSION_STAY1	Frame.Command = *Session* AND BytesToBeSent <> 0 AND IS_CRC_CORRECT(Frame, LastCrc, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE) = TRUE	LastCrc := Frame.Crc0; SendFrame(Session, ADR(SessionId '2-BytesToBeSent'), Frame. Crc0, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE); LastCrc := SendFrame.Crc0; BytesToBeSent := UPDATE_BYTES_ TO_BE_SENT(BytesToBeSent); SecondSessionFrameSent := TRUE; START_WD(SafePara.Watchdog);	Session (Сеанс)
SESSION_ RESET1	Frame.Command = *Reset*	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; SessionId := CREATE_SESSION_ID(); DataCommand := *FailSafeData*; SendFrame(Session, ADR(SessionId), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); LastCrc = SendFrame.Crc0; BytesToBeSent := UPDATE_BYTES_ TO_BE_SENT(2); START_WD(SafePara.Watchdog);	Session (Сеанс)
SESSION_FAIL3	Frame.Command = *Connection* OR Frame.Command = *Parameter* OR Frame.Command = *ProcessData* OR Frame.Command = *FailSafeData*	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CMD; SendFrame(Reset, ADR(CommfaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_ WD(SafePara.Watchdog);	Reset (Сброс)

Переход	Условие	Действие	Следующее состояние
SESSION_FAIL4	Frame.Command <> *Reset* AND Frame.Command <> *Session* AND Frame.Command <> *Connection* AND Frame.Command <> *Parameter* AND Frame.Command <> *ProcessData* AND Frame.Command <> *FailSafeData*	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := UNKNOWN_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE) MasterSeqNo := 1; START_ WD(SafePara.Watchdog);	Reset (Сброс)

7.4.3.2 Событие истекшего сторожевого таймера

Переход	Условие	Действие	Следующее состояние
SESSION_WD		LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := FailSafeData; CommFaultReason := WD_EXPIRED; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara. Watchdog);	Reset (Сброс)

7.4.3.3 Событие сброса соединения

Переход	Условие	Действие	Следующее состояние
SESSION_RESET2		LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := FailSafeData; CommFaultReason := 0; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara.Watchdog);	Reset (Сброс)

7.4.3.4 Событие Команды Set Data

Переход	Условие	Действие	Следующее состояние
SESSION_STAY2		DataCommand := DataCmd;	Session (Сеанс)

7.4.4 Состояние соединения

7.4.4.1 Событие получения кадра

Переход	Условие	Действие	Следующее состояние
CONN_OK	Frame.Command = *Connection* AND BytesToBeSent = 0 AND Frame.ConnId = ConnData. ConnId AND IS_SAFEDATA_CORRECT(Frame, ADR(ConnData), 4-BytesToBeSent) = TRUE AND IS_CRC_CORRECT(Frame, LastCrc, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE) = TRUE	LastCrc := Frame.Crc0; SendFrame(Parameter, ADR(SafePara), Frame.Crc0, ConnData. ConnId, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE); LastCrc := SendFrame.Crc0; BytesToBeSent := UPDATE_ BYTES_TO_BE_SENT(SafeParaSize); START_ WD(SafePara.Watchdog);	Parameter (Параметры)
CONN_FAIL1	Frame.Command = *Connection* AND Frame.ConnId = ConnData. ConnId AND IS_SAFEDATA_CORRECT(Frame, ADR(ConnData), 4-BytesToBeSent) = TRUE AND IS_CRC_CORRECT(Frame, LastCrc, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE) = FALSE	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_ CRC; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_ WD(SafePara.Watchdog);	Reset (Сброс)
CONN_FAIL2	Frame.Command = *Connection* AND Frame.ConnId = ConnData.ConnId AND IS_SAFEDATA_CORRECT(Frame, ADR(ConnData), 4-BytesToBeSent) = FALSE	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_ DATA; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_ WD(SafePara.Watchdog);	Reset (Сброс)

Переход	Условие	Действие	Следующее состояние
CONN_FAIL3	Frame.Command = Connection AND Frame.ConnId <> ConnData. ConnId	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_ CONNID; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_ WD(SafePara.Watchdog);	Reset (Сброс)
CONN_STAY1	Frame.Command = *Connection* AND BytesToBeSent <> 0 AND Frame.ConnId = ConnData. ConnId AND IS_SAFEDATA_CORRECT(Frame, ADR(ConnData), 4-BytesToBeSent) = TRUE AND IS_CRC_CORRECT(Frame, LastCrc, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE) = TRUE	LastCrc := Frame.Crc0; SendFrame(Connection, ADR(ConnData[4- BytesToBeSent]), Frame.Crc0, ConnData. ConnId, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE); LastCrc := SendFrame.Crc0; BytesToBeSent := UPDATE_ BYTES_TO_BE_SENT(BytesToBeSent); START_ WD(SafePara.Watchdog);	Connection (Соединение)
CONN_RESET1	Frame.Command = Reset	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; SessionId := CREATE_SESSION_ ID(); SendFrame(Session, ADR(SessionId), LastCRC, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); LastCrc = SendFrame.Crc0 BytesToBeSent := UPDATE_ BYTES_TO_BE_SENT(2); START_WD(SafePara.Watchdog);	Session (Сеанс)

Переход	Условие	Действие	Следующее состояние
CONN_FAIL4	Frame.Command = *Session* OR Frame.Command = *Parameter* OR Frame.Command = *ProcessData* OR Frame.Command = *FailSafeData*	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara.Watchdog);	Reset (Сброс)
CONN_FAIL5	Frame.Command <> *Reset* AND Frame.Command <> *Session* AND Frame.Command <> *Connection* AND Frame.Command <> *Parameter* AND Frame.Command <> *ProcessData* AND Frame.Command <> *FailSafeData*	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := UNKNOWN_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara.Watchdog);	Reset (Сброс)

7.4.4.2 Событие истекшего сторожевого таймера

Переход	Условие	Действие	Следующее состояние
CONN_WD		LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := FailSafeData; CommFaultReason := WD_EXPIRED; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara.Watchdog);	Reset (Сброс)

7.4.4.3 Событие сброса соединения

Переход	Условие	Действие	Следующее состояние
CONN_RESET2		LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := FailSafeData; CommFaultReason := 0; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara.Watchdog);	Reset (Сброс)

7.4.4.4 Событие Команды Set Data

Переход	Условие	Действие	Следующее состояние
CONN_STAY2		DataCommand := DataCmd;	Connection (Соединение)

7.4.5 Состояние параметров (параметрическое)

7.4.5.1 Событие получения кадра

Переход	Условие	Действие	Следующее состояние
PARA_OK	Frame.Command = Parameter AND BytesToBeSent = 0 AND Frame.ConnId = ConnData. ConnId AND IS_SAFEDATA_CORRECT(Frame, ADR(SafePara), SafeParaSize- BytesToBeSent) = TRUE AND IS_CRC_CORRECT(Frame, LastCrc, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE) = TRUE	LastCrc := Frame.Crc0; SendFrame(DataCommand, ADR(SafeOutputs), Frame. Crc0, ConnData.ConnId, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE); LastCrc := SendFrame.Crc0; START_WD(SafePara.Watchdog);	Data (Данные)
PARA_FAIL1	Frame.Command = Parameter AND Frame.ConnId = ConnData. ConnId AND IS_SAFEDATA_CORRECT(Frame, ADR(SafePara), SafeParaSize- BytesToBeSent) = TRUE AND IS_CRC_CORRECT(Frame, LastCrc, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE) = FALSE	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := FailSafeData; CommFaultReason := INVALID_CRC; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_ WD(SafePara.Watchdog);	Reset (Сброс)

Переход	Условие	Действие	Следующее состояние
PARA_FAIL2	Frame.Command = *Parameter* AND Frame.ConnId = ConnData.ConnId AND IS_SAFEDATA_CORRECT(Frame, ADR(SafePara), SafeParaSize-BytesToBeSent) = FALSE	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_DATA; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara.Watchdog);	Reset (Сброс)
PARA_FAIL3	Frame.Command = *Parameter* AND Frame.ConnId <> ConnData.ConnId	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CONNID; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara.Watchdog);	Reset (Сброс)
PARA_STAY1	Frame.Command = *Parameter* AND BytesToBeSent <> 0 AND Frame.ConnId = ConnData.ConnId AND IS_SAFEDATA_CORRECT(Frame, ADR(SafePara), SafeParaSize-BytesToBeSent) = TRUE AND IS_CRC_CORRECT(Frame, LastCrc, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE) = TRUE	LastCrc := Frame.Crc0; SendFrame(Parameter, ADR(SafePara[SafeParaSize-BytesToBeSent]), Frame.Crc0, ConnData.ConnId, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE); LastCrc := SendFrame.Crc0; BytesToBeSent := UPDATE_BYTES_TO_BE_SENT(BytesToBeSent); START_WD(SafePara.Watchdog);	Parameter (Параметры)

Переход	Условие	Действие	Следующее состояние
PARA_RESET1	Frame.Command = *Reset*	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; SessionId := CREATE_SESSION_ID(); SendFrame(Session, ADR(SessionId), LastCRC, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); LastCrc = SendFrame.Crc0 BytesToBeSent := UPDATE_BYTES_TO_BE_SENT(2); START_WD(SafePara.Watchdog);	Session (Сеанс)
PARA_FAIL4	Frame.Command = *Session* OR Frame.Command = *Connection* OR Frame.Command = *ProcessData* OR Frame.Command = *FailSafeData*	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara.Watchdog);	Reset (Сброс)
PARA_FAIL5	Frame.Command <> *Reset* AND Frame.Command <> *Session* AND Frame.Command <> *Connection* AND Frame.Command <> *Parameter* AND Frame.Command <> *ProcessData* AND Frame.Command <> *FailSafeData*	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := UNKNOWN_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara.Watchdog);	Reset (Сброс)

7.4.5.2 Событие истекшего сторожевого таймера

Переход	Условие	Действие	Следующее состояние
PARA_WD		LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := FailSafeData; CommFaultReason := WD_EXPIRED; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara. Watchdog);	Reset (Сброс)

7.4.5.3 Событие сброса соединения

Переход	Условие	Действие	Следующее состояние
PARA_RESET2		LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := FailSafeData; CommFaultReason := 0; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara.Watchdog);	Reset (Сброс)

7.4.5 Событие команды Set Data

Переход	Условие	Действие	Следующее состояние
PARA_STAY2		DataCommand := DataCmd;	Parameter (Параметры)

7.4.6 Состояние данных

7.4.6.1 Событие получения кадра

Переход	Условие	Действие	Следующее состояние
DATA_OK1	Frame.Command = *ProcessData* AND Frame.ConnId = ConnData.ConnId AND IS_CRC_CORRECT(Frame, LastCrc, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE) = TRUE	SafeInputs := Frame.SafeData; LastCrc := Frame.Crc0; SendFrame(DataCommand, ADR(SafeOutputs), Frame.Crc0, ConnData.ConnId, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE); LastCrc := SendFrame.Crc0; START_WD(SafePara.Watchdog);	Data (Данные)
DATA_OK2	Frame.Command = *FailSafeData* AND Frame.ConnId = ConnData.ConnId AND IS_CRC_CORRECT(Frame, LastCrc, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE) = TRUE	SafeInputs := FS_VALUE; LastCrc := Frame.Crc0; SendFrame(DataCommand, ADR(SafeOutputs), Frame.Crc0, ConnData.ConnId, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE); LastCrc := SendFrame.Crc0; START_WD(SafePara.Watchdog);	Data (Данные)
DATA_FAIL1	(Frame.Command = *ProcessData* OR Frame.Command = *FailSafeData*) AND Frame.ConnId = ConnData.ConnId AND IS_CRC_CORRECT(Frame, LastCrc, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE) = FALSE	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; SafeInputs := FS_VALUE; CommFaultReason := INVALID_CRC; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara.Watchdog);	Reset (Сброс)
DATA_FAIL2	(Frame.Command = *ProcessData* OR Frame.Command = *FailSafeData*) AND Frame.ConnId <> ConnData.ConnId	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; SafeInputs := FS_VALUE; CommFaultReason := INVALID_CONNID SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara.Watchdog);	Reset (Сброс)

Переход	Условие	Действие	Следующее состояние
DATA_ RESET1	Frame.Command = *Reset*	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; SafeInputs := FS_VALUE; SessionId := CREATE_SESSION_ID(); SendFrame(Session, ADR(SessionId), LastCRC, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); LastCrc = SendFrame.Crc0 BytesToBeSent := UPDATE_BYTES_TO_ BE_SENT(2); START_WD(SafePara.Watchdog);	Session (Сеанс)
DATA_FAIL3	Frame.Command = Session OR Frame.Command = Connection OR Frame.Command = Parameter	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := FailSafeData; CommFaultReason := INVALID_ CMD; SafeInputs := FS_VALUE; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_ WD(SafePara.Watchdog);	Reset (Сброс)
DATA_FAIL4	Frame.Command <> Reset AND Frame.Command <> Session AND Frame.Command <> Connection AND Frame.Command <> Parameter AND Frame.Command <> ProcessData AND Frame.Command <> FailSafeData	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := FailSafeData; SafeInputs := FS_VALUE; CommFaultReason := UNKNOWN_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara.Watchdog);	Reset (Сброс)

7.4.6.2 Событие истекшего сторожевого таймера

Переход	Условие	Действие	Следующее состояние
DATA_WD		LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := FailSafeData; SafeInputs := PS_VALUE; CommFaultReason := WD_EXPIRED SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara. Watchdog);	Reset (Сброс)

7.4.6.3 Событие сброса соединения

Переход	Условие	Действие	Следующее состояние
DATA_RESET2		LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := FailSafeData; SafeInputs := PS_VALUE; CommFaultReason := 0; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE); MasterSeqNo := 1; START_WD(SafePara. Watchdog);	Reset (Сброс)

7.4.6.4 Событие Команды Set Data

Переход	Условие	Действие	Следующее состояние
DATA_STAY		DataCommand := DataCmd;	Data (Данные)

7.5 Таблица состояний для ведомого устройства FSoE

7.5.1 Машина состояний для ведомого устройства FSoE

7.5.1.1 Обзор

В зависимости от коммуникационной процедуры, ведомое устройство FSoE может находиться в состояниях, перечисленных в таблице 34.

Таблица 34 — Состояния ведомого устройства FSoE

Состояние	Описание
Сброс	Соединение FSoE сброшено (выводы в безопасном состоянии)
Сеанс	Передается ID сеанса (выводы в безопасном состоянии)
Соединение	Передается ID соединения (выводы в безопасном состоянии)

Окончание таблицы 34

Состояние	Описание
Параметры	Передаются параметры (выводы в безопасном состоянии)
Данные	Передаются данные процесса или отказоустойчивые данные (выводы активны, только если получена команда *ProcessData*)

Диаграмма состояний для ведомого устройства FSoE показана на рисунке 10.

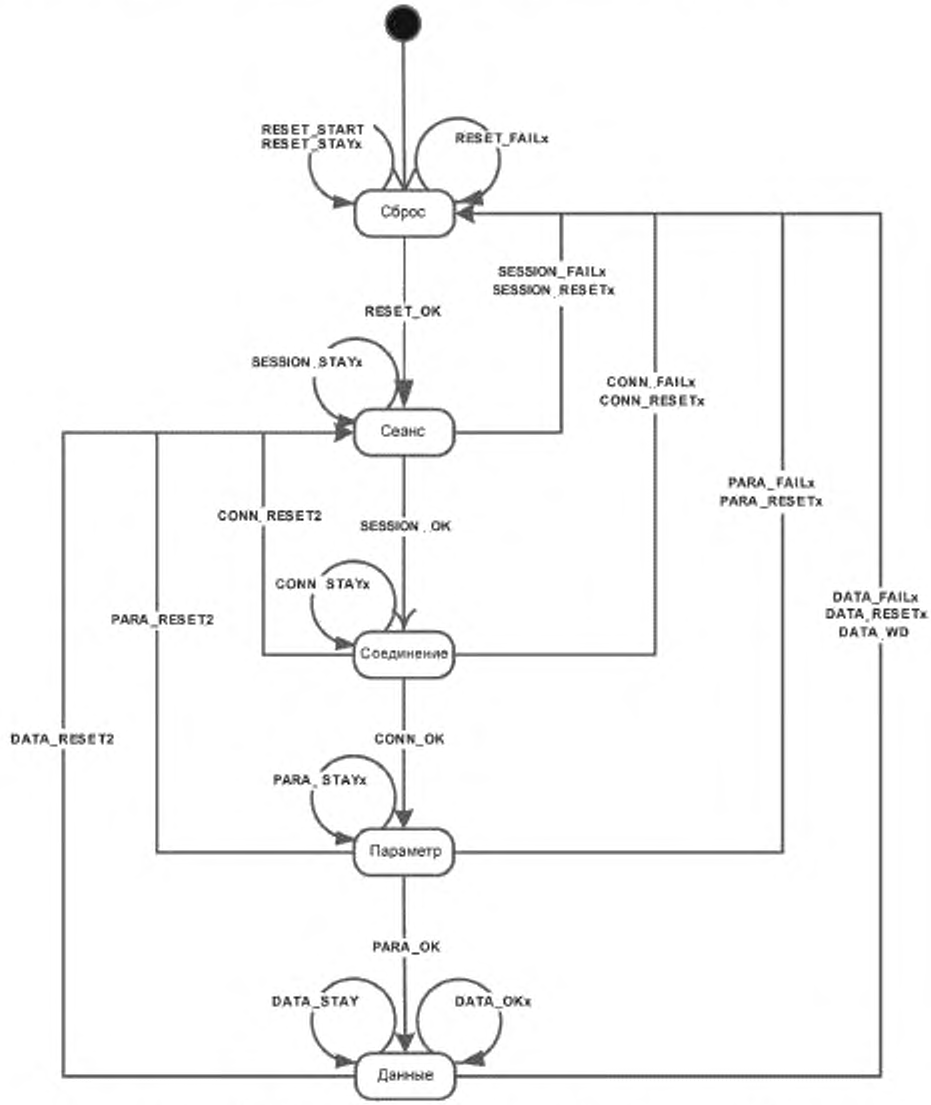


Рисунок 10 — Диаграмма состояний для ведомого устройства FSoE

В следующих секциях проводится анализ событий, которые могут произойти на ведомом устройстве FSoE для каждого состояния.

Каждое событие рассматривается в условиях различных действий или проистекающих состояний.

7.5.1.2 События

Событие может включать в себя различные параметры, на которые приводятся ссылки в таблицах состояний. В таблице 35 приведен список используемых событий.

Таблица 35 — События в таблице состояний ведомого устройства FSoE

Событие	Описание
Получен кадр	Был получен PDU безопасности, т. е. хотя бы один бит в PDU безопасности был изменен . Параметры: Frame — полученный PDU безопасности; Frame.Command — команда полученного PDU безопасности; Frame.Crc0 — CRC_0 полученного PDU безопасности; Frame.ConnId — ID соединения полученного PDU безопасности; Frame.SafeData — данные безопасности полученного PDU безопасности
Истек сторожевой таймер	Истек сторожевой таймер FSoE, т.е. за время сторожевого таймера не было получено никаких PDU безопасности. Параметры: нет
Сброс Соединения	Запрос посредством локального интерфейса на сброс Соединения FSoE. Параметры: нет
Команда Set Data	Запрос посредством локального интерфейса на переключение SafeInputs в состояние безопасности или на выход из состояния безопасности. Параметры: DataCmd — *FailSafeData* или *ProcessData*

7.5.1.3 Действия

В зависимости от разных условий выполняются определенные действия, если происходит событие. В таблицах состояний действия показаны в виде вызовов функций или присваиваний переменных.

В таблице 36 перечислены функции, используемые в таблице состояний ведомого устройства FSoE.

Таблица 36 — Функции в таблице состояний ведущего устройства FSoE

Функция	Описание
SendFrame(cmd, safeData, lastCrc, connId, seqNo, oldCrc, bNew)	Отправлен кадр ведомого устройства FSoE. Параметры: Cmd — команда кадра; SafeData — ссылка на данные безопасности, отправленные с кадром; lastCrc — CRC_0 последнего PDU ведущего устройства безопасности, включенный в вычисление CRC для кадра; connId — ID соединения, которое необходимо ввести в кадр и включить в вычисление CRC; seqNo — указатель на порядковый номер ведомого устройства, включенный в вычисление CRC для кадров. Возвращается приращённый (возможно неоднократно) seqNo; oldCrc — указатель на CRC_0 последнего отправленного PDU ведомого устройства безопасности. Возвращается вычисленный CRC_0.; bNew — если bNew = TRUE и oldCrc равен вычисленному crc, то вычисление CRC повторяется с приращенным seqNo до тех пор, пока вычисленный crc не станет равен oldCrc (процедура соответствует 7.1.3.4)

В таблице 37 перечислены переменные, используемые в таблице состояний ведомого устройства FSoE.

Таблица 37 — Переменные, используемые в таблице состояний ведомого устройства FSoE

Переменная	Описание
LastCrc	CRC_0 последнего PDU ведомого устройства безопасности (инициализируется значением 0 при включении питания)
OldMasterCrc	CRC_0 последнего полученного PDU ведущего устройства безопасности (инициализируется значением 0 при включении питания)

Окончание таблицы 36

Переменная	Описание
OldSlaveCrc	CRC_0 последнего отправленного PDU ведомого устройства безопасности (инициализируется значением 0 при включении питания)
MasterSeqNo	Ожидаемый порядковый номер ведущего устройства для использования в CRC для следующего PDU ведущего устройства безопасности (инициализируется значением 0 при включении питания)
SlaveSeqNo	Порядковый номер ведомого устройства для использования в CRC следующего PDU ведомого устройства безопасности (инициализируется значением 0 при включении питания)
InitSeqNo	Переменная, содержащая порядковый номер инициализации 1
DataCommand	Указывает на то, какая из команд *ProcessData* или *FailSafeData* отправлена в состоянии Данных. Инициализируется с помощью *FailSafeData* при включении питания
BytesToBeSent	Если несколько PDU блоков безопасности должно быть отправлено в состоянии сеанса, соединения или параметров, то эта переменная указывает на то, сколько еще октетов должно быть отправлено (инициализируется значением 0 при включении питания)
ConnectionId	В состоянии соединения ConnectionID принимается ведущим устройством FSoE (инициализируется значением 0 при включении питания)
ConnectionData	В состоянии соединения ConnectionData принимается ведущим устройством FSoE (инициализируется значением 0 при включении питания)
SlaveAddress	Адрес ведомого устройства FSoE инициализируется посредством локального интерфейса при включении питания (как правило, внешний переключатель адресов)
SafePara	SafePara принимаются ведущим устройством FSoE в состоянии параметров и инициализируется в зависимости от устройства. SafePara.Watchdog: сторожевой таймер FSoE (инициализируется значением 0 при включении питания)
ExpectedSafeParaSize	Указывает на длину ожидаемого SafePara
SafeOutputs	Содержит значения процесса выводов безопасности, полученные ведущим устройством FSoE. Инициализируется с помощью FS_VALUE (Fail-safe Data = 0) при включении питания
SafeInputs	Содержит значения процесса вводов безопасности, отправленных ведущему устройству FSoE. Инициализируется с помощью FS_VALUE (Fail-safe Data = 0) при включении питания
CommFaultReason	Указывает на код ошибки в случае события возникновения коммуникационной ошибки

7.5.1.4 Макрокоманды

Определенные функциональные возможности объединяются в макрокоманды для того, чтобы сохранять прозрачность таблиц состояний.

В таблице 38 перечислены макрокоманды из таблицы состояний ведомого устройства FSoE.

Таблица 38 — Макрокоманды в таблице состояний ведомого устройства FSoE

Функция	Описание
IS_CRC_CORRECT(frame, lastCrc, seqNo, oldCrc, bNew)	Эта макрокоманда проверяет, корректны ли CRC коды полученного PDU ведущего устройства безопасности. Параметры: Frame — принятый кадр; lastCrc — CRC_0 последнего отправленного PDU ведомого устройства безопасности, включенное в вычисления CRC для принятого PDU; seqNo — указатель на порядковый номер ведущего устройства, используемый в вычислении CRC для принятого кадра. Возвращается приращенный (возможно неоднократно) seqNo; oldCrc — указатель на CRC_0 последнего принятого PDU ведущего устройства безопасности. Возвращается CRC_0 полученной телеграммы; bNew — если bNew = TRUE и oldCrc равняется вычисленному crc, то вычисление CRC повторяется с приращенным seqNo до тех пор, пока вычисленный crc не перестанет быть равным oldCrc (процедура выполняется согласно 7.1.3.4)

Окончание таблицы 38

Функция	Описание
UPDATE_BYTES_TO_BE_SENT (bytesSent)	Данная макрокоманда проверят, сколько еще октетов в состояниях сеанса, соединения и параметров необходимо отправить перед изменением состояния. Параметры: safePara: указатель на полученные SafePara
IS_SAFE_PARA_CORRECT (safePara)	Данная макрокоманда проверяет корректны ли SafePara полученные в состоянии параметров. Параметры: Frame — принятый кадр; expectedData — ссылка на ожидаемые данные; bytesSent — число отправленных октетов
STORE_DATA(dst, src)	Данная макрокоманда хранит полученные данные безопасности PDU безопасности. Параметры: Dst — указатель на целевой адрес; Src — указатель на адрес источника
GET_PARA_FAULT()	Данная макрокоманда возвращает код ошибки, если SafePara не действителен
START_WD (watchdog)	Данная макрокоманда перезапускает (сбрасывает) сторожевой таймер и запускает контролирующий таймер с указанным сторожевым таймером. Параметры: Watchdog — время контроля (мониторинга) в мс
STOP_WD()	Данная макрокоманда останавливает контролирующий таймер и сбрасывает сторожевой таймер
ADR	Данная макрокоманда генерирует ссылку (указатель) на переменную

7.5.2 Состояние сброса

7.5.2.1 Событие получения кадра

Переход	Условие		Следующее состояние

RESET_OK	`Frame.Command = Session AND IS_CRC_CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE) = TRUE`	`LastCrc := Frame.Crc0;`
		`SessionId := CREATE_SESSION_ID();`
		`SendFrame(Session, ADR(SessionId), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE);`
		`LastCrc := SendFrame.Crc0;`
		`BytesToBeSent := UPDATE_BYTES_TO_BE_SENT(2);`
			Session (Сеанс)

RESET_FAIL1	`Frame.Command = Session AND IS_CRC_CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE) = FALSE`	`LastCrc := 0`
		`OldMasterCrc := 0;`
		`OldSlaveCrc := 0;`
		`MasterSeqNo := 1;`
		`SlaveSeqNo := 1;`
		`DataCommand := FailSafeData;`
		`CommFaultReason := INVALID_CRC;`
		`SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE);`
		`SlaveSeqNo := 1;`
			Reset (Сброс)

Переход	Условие		Следующее состояние
RESET_STAY1	Frame.Command = *Reset*	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; InitSeqNo := 1; DataCommand := FailSafeData; CommFaultReason := 0; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
RESET_FAIL2	(Frame.Command = *Connection* OR Frame.Command = *Parameter* OR Frame.Command = ProcessData OR Frame.Command = FailSafeData)	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
RESET_FAIL3	(Frame.Command <> *Reset* AND Frame.Command <> *Session* AND Frame.Command <> *Connection* AND Frame.Command <> *Parameter* AND Frame.Command <> ProcessData AND Frame.Command <> FailSafeData)	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := UNKNOWN_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

7.5.2.2 Событие истекшего сторожевого таймера

Невозможно в данном состоянии, так как сторожевой таймер еще не был запущен.

7.5.2.3 Событие сброса соединения

Переход	Условие	Действие	Следующее состояние
RESET_START	Frame.Command = *Session* AND IS_CRC_CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), FALSE) = TRUE	LastCrc := 0 OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; InitSeqNo := 1; DataCommand := FailSafeData; CommFaultReason := 0; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

7.5.2.4 Событие Команды Set Data

Переход	Условие	Действие	Следующее состояние
RESET_STAY2		DataCommand := DataCmd;	Reset (Сброс)

7.5.3 Состояние сеанса

7.5.3.1 Событие получения кадра

Переход	Условие	Действие	Следующее состояние
SESSION_OK		STORE_DATA(ADR(ConnectionData), ADR(Frame.SafeData)); ConnectionId := Frame.ConnId; LastCrc := Frame.Crc0; SendFrame(*Connection*, ADR(Frame.SafeData), LastCrc, ConnectionId, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE); LastCrc := SendFrame.Crc0; BytesToBeSent := UPDATE_BYTES_TO_BE_SENT(4);	Connection (Соединение)
SESSION_FAIl1	Frame.Command = *Connection* AND BytesToBeSent = 0 AND Frame.ConnId <> 0 AND IS_CRC_CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE) = FALSE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CRC; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

Переход	Условие	Действие	Следующее состояние
SESSION_FAIL2	Frame.Command = *Connection* AND BytesToBeSent = 0 AND Frame.ConnId = 0	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CONNID; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
SESSION_FAIL3	Frame.Command = *Connection* AND BytesToBeSent <> 0	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1 DataCommand := *FailSafeData*; CommFaultReason := INVALID_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
SESSION_STAY1	Frame.Command = *Session* AND BytesToBeSent <> 0 AND IS_CRC_ CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE) = TRUE	LastCrc := Frame.Crc0; SendFrame(Session, ADR(SessionId[2- BytesToBeSent]), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE); LastCrc := SendFrame.Crc0; BytesToBeSent := UPDATE_BYTES_TO_ BE_SENT(BytesToBeSent);	Session (Сеанс)
SESSION_STAY2	Frame.Command = *Session* AND IS_CRC_CORRECT(Frame, 0, ADR(InitSeqNo), ADR(OldMasterCrc), FALSE) = TRUE	LastCrc := Frame.Crc0; MasterSeqNo := InitSeqNo; InitSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; SessionId := CREATE_SESSION_ID(); SendFrame(Session, ADR(SessionID), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); LastCrc := SendFrame.Crc0; BytesToBeSent := UPDATE_BYTES_TO_ BE_SENT(2);	Session (Сеанс)

Переход	Условие	Действие	Следующее состояние
SESSION_FAIL4[a)]	Frame.Command = *Session* AND IS_CRC_CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE) = FALSE AND IS_CRC_ CORRECT(Frame, 0, ADR(InitSeqNo), ADR(OldMasterCrc), FALSE) = FALSE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; InitSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CRC; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
SESSION_FAIL5[a)]	Frame.Command = *Session* AND BytesToBeSent = 0 AND IS_CRC_CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE) = TRUE AND IS_CRC_ CORRECT(Frame, 0, ADR(InitSeqNo), ADR(OldMasterCrc), FALSE) = FALSE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CME; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
SESSION_ RESET1	Frame.Command = *Reset* AND IS_CRC_ CORRECT(Frame, 0, ADR(InitSeqNo), ADR(OldMasterCrc), FALSE) = TRUE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; InitSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := 0; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE);	Reset (Сброс)
SESSION_FAIL6	Frame.Command = *Reset* AND IS_CRC_ CORRECT(Frame, 0, ADR(InitSeqNo), ADR(OldMasterCrc), FALSE) = FALSE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; InitSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CRC; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

Переход	Условие	Действие	Следующее состояние
SESSION_FAIL7	Frame.Command = *Parameter* OR Frame.Command = *ProcessData* OR Frame.Command = *FailSafeData*	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
SESSION_FAIL8	(Frame.Command <> *Reset* AND Frame.Command <> *Session* AND Frame.Command <> *Connection* AND Frame.Command <> *Parameter* AND Frame.Command <> *ProcessData* AND Frame.Command <> *FailSafeData*)	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := UNKNOWN_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

[a] Два состояния SESSION_FAIL4 и SESSION_FAIL5 являются единственными состояниями, в которых должны вычисляться две проверки CRC. Единственное отличие заключается в другом CommFaultReason, т. е. только диагностическая информация, не информация важная для безопасности. Разрешается сокращать эти состояния до одного; в этом состоянии только условие «IS_CRC_CORRECT(Frame, 0, ADR(InitSeqNo), ADR(OldMasterCrc), FALSE) = FALSE» должно проверяться с помощью CommFaultReason := INVALID_CRC.

7.5.3.2 Событие истекшего сторожевого таймера

Невозможно в данном состоянии, так как сторожевой таймер еще не был запущен.

7.5.3.3 Событие сброса соединения

Переход	Условие	Действие	Следующее состояние
SESSION_RESET2		LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := FailSafeData; CommFaultReason := 0; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

7.5.3.4 Событие Команды Set Data

Переход	Условие	Действие	Следующее состояние
SESSION_STAY3		DataCommand := DataCmd;	Session (Сеанс)

7.5.4 Состояние соединения

7.5.4.1 Событие получения кадра

Переход	Условие	Действие	Следующее состояние
CONN_OK	Frame.Command = *Parameter* AND BytesToBeSent = 0 AND Frame.ConnId = ConnectionId AND ConnectionData. ConnectionId = ConnectionId AND ConnectionData. SlaveAddress = SlaveAddress AND IS_CRC_CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE) = TRUE	STORE_DATA(ADR(SafePara), ADR(Frame.SafeData)); LastCrc := Frame.Crc0; SendFrame(*Parameter*, ADR(Frame.SafeData), LastCrc, ConnectionId, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE); LastCrc := SendFrame.Crc0; BytesToBeSent := UPDATE_BYTES_ TO_BE_SENT(ExpectedSafeParaSize);	Parameter (Параметры)
CONN_FAIL1	Frame.Command = *Parameter* AND BytesToBeSent = 0 AND Frame.ConnId = ConnectionId AND ConnectionData. ConnectionId = ConnectionId AND ConnectionData. SlaveAddress = SlaveAddress AND IS_CRC_CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE) = FALSE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CRC; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

Переход	Условие	Действие	Следующее состояние
CONN_FAIL2	Frame.Command = *Parameter* AND BytesToBeSent = 0 AND Frame.ConnId = ConnectionId AND ConnectionData. ConnectionId = ConnectionId AND ConnectionData. SlaveAddress <> SlaveAddress	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_ ADDR; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
CONN_FAIL3	Frame.Command = *Parameter* AND BytesToBeSent = 0 AND (Frame.ConnId <> ConnectionId OR ConnectionData. ConnectionId <> ConnectionId)	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_ CONNID; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
CONN_FAIL4	Frame.Command = *Parameter* AND BytesToBeSent <> 0	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

Переход	Условие	Действие	Следующее состояние
CONN_STAY1	Frame.Command = *Connection* AND BytesToBeSent <> 0 AND Frame.ConnId = ConnectionId AND IS_CRC_CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE) = TRUE	STORE_DATA(ADR(Connection[4- BytesToBeSent]), ADR(Frame.SafeData)); LastCrc := Frame.Crc0; SendFrame(Connection, ADR(Frame.SafeData), LastCrc, ConnectionId, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE); LastCrc := SendFrame.Crc0; BytesToBeSent := UPDATE_BYTES_ TO_BE_SENT(BytesToBeSent);	Connection (Соединение)
CONN_FAIL5	Frame.Command = *Connection* AND BytesToBeSent <> 0 AND Frame.ConnId = ConnectionId AND IS_CRC_CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE) = FALSE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CRC; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
CONN_FAIL6	Frame.Command = *Connection* AND BytesToBeSent <> 0 AND Frame.ConnId <> ConnectionId	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_ CONID; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

Переход	Условие	Действие	Следующее состояние
CONN_FAIL7	Frame.Command = *Connection* AND BytesToBeSent = 0	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
CONN_RESET1	Frame.Command = *Reset* AND IS_CRC_CORRECT(Frame, 0, ADR(InitSeqNo), ADR(OldMasterCrc), FALSE) = TRUE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; InitSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := 0; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
CONN_FAIL8	Frame.Command = *Reset* AND IS_CRC_CORRECT(Frame, 0, ADR(InitSeqNo), ADR(OldMasterCrc), FALSE) = FALSE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; InitSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CRC; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

Переход	Условие	Действие	Следующее состояние
CONN_RESET2	Frame.Command = *Session* AND IS_CRC_CORRECT(Frame, 0, ADR(InitSeqNo), ADR(OldMasterCrc), FALSE) = TRUE	LastCrc := Frame.Crc0; MasterSeqNo := 2; InitSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; SessionId := CREATE_SESSION_ ID(); SendFrame(Session, ADR(SessionID), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); LastCrc := SendFrame.Crc0; BytesToBeSent := UPDATE_BYTES_ TO_BE_SENT(2);	Session (Сеанс)
CONN_FAIL9	Frame.Command = *Session* AND IS_CRC_CORRECT(Frame, 0, ADR(InitSeqNo), ADR(OldMasterCrc), FALSE) = FALSE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; InitSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CRC; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
CONN_FAIL10	Frame.Command = *ProcessData* OR Frame.Command = *FailSafeData*	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

Переход	Условие	Действие	Следующее состояние
CONN_FAIL11	(Frame.Command <> *Reset* AND Frame.Command <> *Session* AND Frame.Command <> *Connection* AND Frame.Command <> *Parameter* AND Frame.Command <> *ProcessData* AND Frame.Command <> *FailSafeData*)	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := UNKNOWN_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

7.5.4.2 Событие истекшего сторожевого таймера

Невозможно в данном состоянии, так как сторожевой таймер еще не был запущен.

7.5.4.3 Событие сброса соединения

Переход	Условие	Действие	Следующее состояние
CONN_RESET3		LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := FailSafeData; CommFaultReason := 0; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

7.5.4.4 Событие Команды Set Data

Переход	Условие	Действие	Следующее состояние
CONN_STAY2		DataCommand := DataCmd;	Connection (Соединение)

7.5.5 Состояние параметров

7.5.5.1 Событие получения кадра

Переход	Условие	Действие	Следующее состояние
PARA_OK1	Frame.Command = *ProcessData* AND BytesToBeSent = 0 AND Frame.ConnId = ConnectionId AND IS_SAFE_PARA_CORRECT(SafePara) = TRUE AND IS_CRC_CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE) = TRUE	Watchdog := SafePara.Watchdog; SafeOutputs := Frame.SafeData; LastCrc := Frame.Crc0; SendFrame(*DataCommand*, ADR(SafeInputs), LastCrc, ConnectionId, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE); LastCrc := SendFrame.Crc0; START_ WD(Watchdog);	Data (Данные)
PARA_OK2	Frame.Command = *FailSafeData* AND BytesToBeSent = 0 AND Frame.ConnId = ConnectionId AND IS_SAFE_PARA_CORRECT(SafePara) = TRUE AND IS_CRC_CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE) = TRUE	Watchdog := SafePara.Watchdog; SafeOutputs := FS_VALUE; LastCrc := Frame.Crc0; SendFrame(*DataCommand*, ADR(SafeInputs), LastCrc, ConnectionId, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE); LastCrc := SendFrame.Crc0; START_ WD(Watchdog);	Data (Данные)
PARA_FAIL1	(Frame.Command = *ProcessData* OR Frame.Command = *FailSafeData*) AND BytesToBeSent = 0 AND Frame.ConnId = ConnectionId AND IS_SAFE_PARA_CORRECT(SafePara) = TRUE AND IS_CRC_CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE) = FALSE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CRC; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
PARA_FAIL2	(Frame.Command = *ProcessData* OR Frame.Command = *FailSafeData*) AND BytesToBeSent = 0 AND Frame.ConnId = ConnectionId AND IS_SAFE_PARA_CORRECT(SafePara) = FALSE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := GET_PARA_FAULT; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ALR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

Переход	Условие	Действие	Следующее состояние
PARA_FAIL3	(Frame.Command = *ProcessData* OR Frame.Command = *FailSafeData*) AND BytesToBeSent = 0 AND Frame.ConnId <> ConnectionId	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CONNID; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
PARA_FAIL4	(Frame.Command = *ProcessData* OR Frame.Command = *FailSafeData*) AND BytesToBeSent <> 0	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
PARA_STAY1	Frame.Command = *Parameter* AND BytesToBeSent <> 0 AND Frame.ConnId = ConnectionId AND IS_CRC_CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE) = TRUE	STORE_DATA(ADR(SafePara[ExpectedSafeParaSize-BytesToBeSent]), ADR(Frame.SafeData)); LastCrc := Frame.Crc0; SendFrame(*Parameter*, ADR(Frame.SafeData), LastCrc, ConnectionId, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE); LastCrc := SendFrame.Crc0; BytesToBeSent := UPDATE_BYTES_TO_BE_SENT(BytesToBeSent);	Parameter (Параметр)
PARA_FAIL5	Frame.Command = *Parameter* AND BytesToBeSent <> 0 AND Frame.ConnId = ConnectionId AND IS_CRC_CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE) = FALSE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CRC; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

Переход	Условие	Действие	Следующее состояние
PARA_FAIL6	Frame.Command = *Parameter* AND BytesToBeSent <> 0 AND Frame.ConnId <> ConnectionId	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CONNID; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
PARA_FAIL7	Frame.Command = *Parameter* AND BytesToBeSent = 0	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
PARA_RESET1	Frame.Command = *Reset* AND IS_CRC_ CORRECT(Frame, 0, ADR(InitSeqNo), ADR(OldMasterCrc), FALSE) = TRUE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; InitSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := 0; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
PARA_FAIL8	Frame.Command = *Reset* AND IS_CRC_ CORRECT(Frame, 0, ADR(InitSeqNo), ADR(OldMasterCrc), FALSE) = FALSE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; InitSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CRC; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

Переход	Условие	Действие	Следующее состояние
PARA_RESET2	Frame.Command = *Session* AND IS_CRC_CORRECT(Frame, 0, ADR(InitSeqNo), ADR(OldMasterCrc), FALSE) = TRUE	LastCrc := Frame.Crc0; MasterSeqNo := 2; InitSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; SessionId := CREATE_SESSION_ID(); SendFrame(Session, ADR(SessionID), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); LastCrc := SendFrame.Crc0; BytesToBeSent := UPDATE_BYTES_TO_BE_SENT(2);	Session (Сеанс)
PARA_FAIL9	Frame.Command = *Session* AND IS_CRC_CORRECT(Frame, 0, ADR(InitSeqNo), ADR(OldMasterCrc), FALSE) = FALSE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; InitSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CRC; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
PARA_FAIL10	Frame.Command = *Connection*	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := INVALID_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

Переход	Условие	Действие	Следующее состояние
PARA_FAIL11	(Frame.Command <> *Reset* AND Frame.Command <> *Session* AND Frame.Command <> *Connection* AND Frame.Command <> *Parameter* AND Frame.Command <> *FailSafeData* AND Frame.Command <> *ProcessData*)	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; CommFaultReason := UNKNOWN_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

7.5.5.2 Событие истекшего сторожевого таймера

Невозможно в данном состоянии, так как сторожевой таймер еще не был запущен.

7.5.5.3 Событие сброса соединения

Переход	Условие	Действие	Следующее состояние
PARA_RESET3		LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := FailSafeData; CommFaultReason := 0; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

7.5.5.4 Событие Команды Set Data

Переход	Условие	Действие	Следующее состояние
PARA_STAY2		DataCommand := DataCmd;	Parameter (Параметры)

7.5.6 Состояние данных

7.5.6.1 Событие получения кадра

Переход	Условие	Действие	Следующее состояние
DATA_OK1	Frame.Command = *ProcessData* AND Frame.ConnId = ConnectionId AND IS_CRC_CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE) = TRUE	SafeOutputs := Frame.SafeData; LastCrc := Frame.Crc0; SendFrame(DataCommand, ADR(SafeInputs), LastCrc, ConnectionId, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE); LastCrc := SendFrame.Crc0; START_WD(Watchdog);	Data (Данные)
DATA_OK2	Frame.Command = *FailSafeData* AND Frame.ConnId = ConnectionId AND IS_CRC_CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE) = TRUE	SafeOutputs := FS_VALUE; LastCrc := Frame.Crc0; SendFrame(DataCommand, ADR(SafeInputs), LastCrc, ConnectionId, ADR(SlaveSeqNo), ADR(OldSlaveCrc), TRUE); LastCrc := SendFrame.Crc0; START_WD(Watchdog);	Data (Данные)
DATA_FAIL1	(Frame.Command = *ProcessData* OR Frame.Command = *FailSafeData*) AND Frame.ConnId = ConnectionId AND IS_CRC_CORRECT(Frame, LastCrc, ADR(MasterSeqNo), ADR(OldMasterCrc), TRUE) = FALSE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; SafeOutputs := FS_VALUE; STOP_WD(); CommFaultReason := INVALID_CRC; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
DATA_FAIL2	(Frame.Command = *ProcessData* OR Frame.Command = *FailSafeData*) AND Frame.ConnId <> ConnectionId	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; SafeOutputs := FS_VALUE; STOP_WD(); CommFaultReason := INVALID_CONNID; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

Переход	Условие	Действие	Следующее состояние
DATA_RESET1	Frame.Command = *Reset* AND IS_CRC_CORRECT(Frame, 0, ADR(InitSeqNo), ADR(OldMasterCrc), FALSE) = TRUE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; InitSeqNo := 1; DataCommand := *FailSafeData*; SafeOutputs := FS_VALUE; STOP_WD(); CommFaultReason := 0; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
DATA_FAIL3	Frame.Command = *Reset* AND IS_CRC_CORRECT(Frame, 0, ADR(InitSeqNo), ADR(OldMasterCrc), FALSE) = FALSE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; InitSeqNo := 1; DataCommand := *FailSafeData*; SafeOutputs := FS_VALUE; STOP_WD(); CommFaultReason := INVALID_CRC; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
DATA_RESET2	Frame.Command = *Session* AND IS_CRC_CORRECT(Frame, 0, ADR(InitSeqNo), ADR(OldMasterCrc), FALSE) = TRUE	LastCrc := Frame.Crc0; MasterSeqNo := 2; InitSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; SafeOutputs := FS_VALUE; STOP_WD(); SessionId := CREATE_SESSION_ID(); SendFrame(Session, ADR(SessionID), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); LastCrc := SendFrame.Crc0; BytesToBeSent := UPDATE_BYTES_TO_BE_SENT(2);	Session (Сеанс)

Переход	Условие	Действие	Следующее состояние
DATA_FAIL4	Frame.Command = *Session* AND IS_CRC_CORRECT(Frame, 0, ADR(InitSeqNo), ADR(OldMasterCrc), FALSE) = FALSE	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; InitSeqNo := 1; DataCommand := *FailSafeData*; SafeOutputs := FS_VALUE; STOP_WD(); CommFaultReason := INVALID_CRC; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
DATA_FAIL5	Frame.Command = *Connection* OR Frame.Command = *Parameter*	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; SafeOutputs := FS_VALUE; STOP_WD(); CommFaultReason := INVALID_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)
DATA_FAIL6	(Frame.Command <> *Reset* AND Frame.Command <> *Session* AND Frame.Command <> *Connection* AND Frame.Command <> *Parameter* AND Frame.Command <> *FailSafeData* AND Frame.Command <> *ProcessData*)	LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := *FailSafeData*; SafeOutputs := FS_VALUE; STOP_WD(); CommFaultReason := UNKNOWN_CMD; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

7.5.6.2 Событие истекшего сторожевого таймера

Переход	Условие	Действие	Следующее состояние
DATA_WD		LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := FailSafeData; SafeOutputs := FS_VALUE; STOP_WD(); CommFaultReason := WD_EXPIRED; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

7.5.6.3 Событие сброса соединения

Переход	Условие	Действие	Следующее состояние
DATA_RESET3		LastCrc := 0; OldMasterCrc := 0; OldSlaveCrc := 0; MasterSeqNo := 1; SlaveSeqNo := 1; DataCommand := FailSafeData; SafeOutputs := FS_VALUE; STOP_WD(); CommFaultReason := 0; SendFrame(Reset, ADR(CommFaultReason), LastCrc, 0, ADR(SlaveSeqNo), ADR(OldSlaveCrc), FALSE); SlaveSeqNo := 1;	Reset (Сброс)

7.5.6.4 Событие Команды Set Data

Переход	Условие	Действие	Следующее состояние
DATA_STAY		DataCommand := DataCmd;	Data (Данные)

8 Управление коммуникационным уровнем безопасности

8.1 Обработка параметров FSCP 12/1

Протокол FSCP 12/1 поддерживает встроенную возможность скачивания параметров ведомого устройства FSoE у ведущего устройства FSoE в состоянии параметров.

8.2 Параметры коммуникаций FSoE

Коммуникации FSoE между ведущим устройством FSoE и ведомым устройством FSoE используют коммуникационные параметры, заданные в таблице 39.

Таблица 39 — FSoE Communication parameters

Название	Тип данных	Диапазон	Описание
ID Сеанса FSoE	UINT16	0 .. 2^{16}	Произвольно генерируемый ID сеанса FSoe
ID соединения FSoE	UINT16	1 ... 2^{16}	Уникальный ID соединения между ведущим устройством FSoE и ведомым устройством FSoE
Порядковый номер FSoE	UINT16	Init: 0 1 ... 2^{16}	Увеличивается в каждом цикле FSoE
Адрес ведомого устройства FSoE	UINT16	1 ... 2^{16}	Уникальный адрес ведомого устройства FSoE для каждого устройства FSoE
Время сторожевого таймера FSoE	UINT16	1 ... 2^{16}	Время сторожевого таймера для соединения FSoE в мс

9 Системные требования

9.1 Индикаторы и коммутаторы

9.1.1 Состояния индикаторов и частота вспышек

Состояния индикаторов и частота вспышек определены в таблице 40 и на рисунке 11. Перечисленные в этих таблицах времена должны соблюдаться с отклонением меньше чем ± 20%.

Таблица 40 — Состояния индикаторов

Состояние индикатора	Определение
включен	Индикатор должен быть постоянно включен
выключен	Индикатор должен быть постоянно выключен
единичная вспышка	Одно короткое мерцание индикатора (50 мс), за которым следует фаза «выключен» длительностью минимум 200 мс
мерцание	Индикатор должен включаться и выключаться через равные интервалы с частотой в 10 Гц, т. е. 50 мс включен и 50 мс выключен
мигание	Индикатор должен включаться и выключаться через равные интервалы с частотой в 2,5 Гц, т. е. 200 мс включен и 200 мс выключен
мерцание с 1 вспышкой	Сначала индикатор должен мерцать 500 мс, затем одна фаза «выключен» (500 мс), затем короткая вспышка (200 мс), за которой следует длинная фаза «выключен» (1 000 мс)
мерцание с 2 вспышками	Сначала индикатор должен мерцать 500 мс, затем одна фаза «выключен» (500 мс), затем последовательность из двух коротких вспышек (200 мс), разделенных фазой «выключен» (200 мс), за которой следует длинная фаза «выключен» (1 000 мс)
мерцание с n вспышками	Сначала индикатор должен мерцать 500 мс, затем одна фаза «выключен» (500 мс), затем последовательность из *n* коротких вспышек (200 мс), разделенных фазой «выключен» (200 мс), за которой следует длинная фаза «выключен» (1 000 мс)

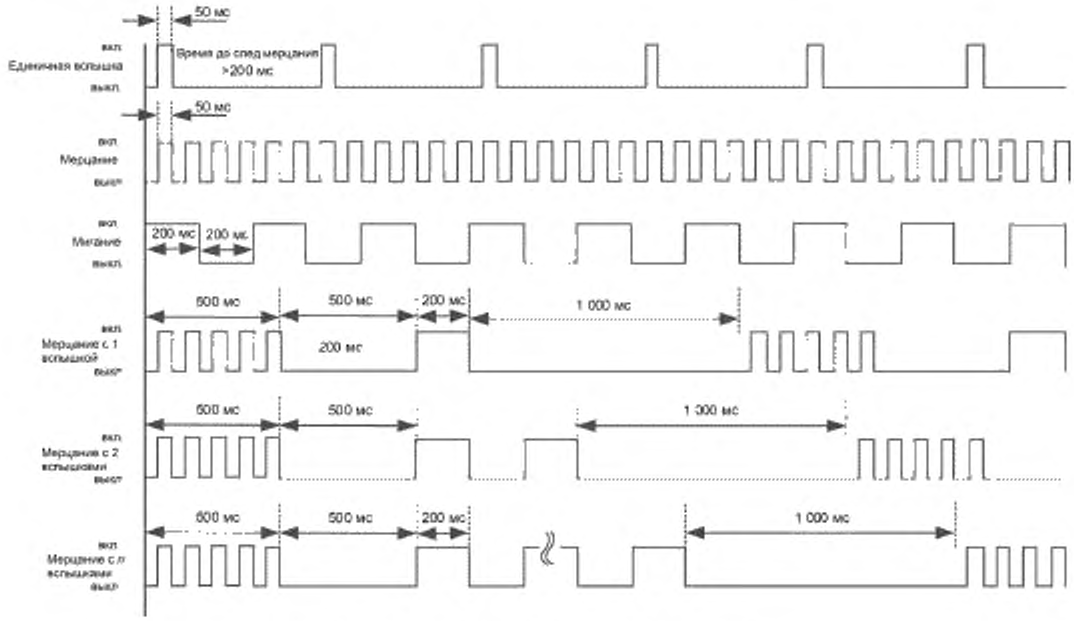


Рисунок 11 — Частота вспышек индикаторов

9.1.2 Индикаторы

9.1.2.1 Требующиеся индикаторы

Устройство, поддерживающее протокол FSCP 12/1 должно иметь индикатор статуса для поддержки при визуальном осмотре и выявлении неисправностей соединения FSoE. Если устройство поддерживает какие-либо из описанных здесь индикаторов, то они должны придерживаться спецификаций.

Могут быть задействованы дополнительные индикаторы.

9.1.2.2 Индикатор FSoE STATUS

Индикатор STATUS (статус) FSoE должен отображать статус FSoE соединения. Он должен гореть зеленым цветом.

Для того чтобы не нарушать пространственные ограничения, можно опустить маркировку индикатора STATUS для FSoE. Если же индикатор промаркирован, то маркировка должна быть следующей (не чувствительна к регистру):

- FS;
- FSoE;
- Статус FSoE.

Состояния индикатора STATUS для FSoE описаны в таблице 41.

Таблица 41 — Состояния индикатора STATUS для FSoE

Состояние индикатора	Определение	Состояние FSoE
выключен	Осуществляется инициализация	Перед сбросом (Pre-Reset)
мигает	Готов к параметризации	Сброс, сеанс, соединение, параметры
включен	Нормальное функционирование	Данные Процесса
единичная вспышка	Отказоустойчивые данные	Отказоустойчивые данные
мерцание	Неустановленная ошибка в соединении FSoE	Все
мерцание с 1 вспышкой	Ошибка в F-параметре	Параметры
мерцание с 2 вспышками	Ошибка в параметре приложения	Параметры

Окончание таблицы 41

Состояние индикатора	Определение	Состояние FSoE
мерцание с 3 вспышками	Неверный адрес безопасности	Соединение
мерцание с 4 вспышками	Неверная команда	Все
мерцание с 5 вспышками	Ошибка сторожевого таймера	Все
мерцание с 6 вспышками	CRC-Ошибка	Все

Индикация ошибки (мерцание) должна длиться до тех пор, пока соединение FSoE не сменит состояние сброс на состояние сеанс.

Должна быть показана хотя бы одна индикация в виде полной последовательности мерцаний с *n* числом вспышек.

9.2 Руководство по установке

В настоящем стандарте описан протокол и услуги для коммуникационной системы безопасности, основанные на МЭК 61158, Тип 12. Тем не менее, применение устройств безопасности вместе с протоколом безопасности, описанным в настоящем стандарте, требует их надлежащую установку. Все устройства, соединенные с коммуникационной системой безопасности, определенной в настоящем стандарте, должны выполнять требования SELV/PELV, описанные в соответствующих стандартах МЭК, таких как МЭК 60204-1. Другие важные руководящие указания по установке приведены в МЭК 61918.

9.3 Время реакции функции безопасности

9.3.1 Общие положения

Для определения времени реакции функции безопасности, функция безопасности декомпозируется на несколько компонентов, как это показано на рисунке 12.

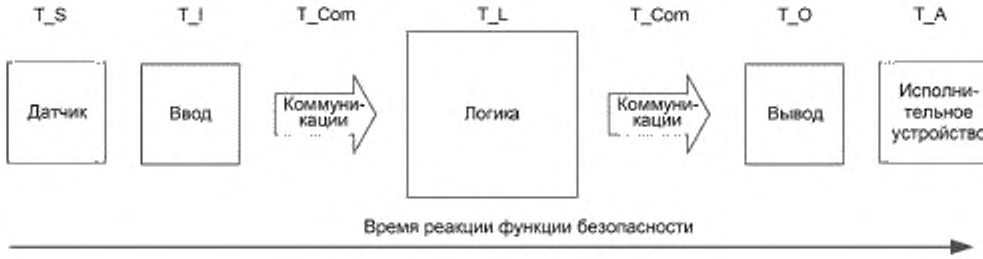


Рисунок 12 — Компоненты функции безопасности

Не все компоненты должны быть представлены в системе. Датчик (например, световая завеса или кнопка аварийной остановки) преобразует физический сигнал в электрический. Этот сигнал может быть подключен к устройству ввода (например, вводу безопасности), который преобразует сигнал в логическую информацию. Эта логическая информация передается по коммуникационной сети безопасности в логический узел безопасности. Логический узел безопасности (например, PLC безопасности) объединяет эту и/или другую входную информацию в логическую выходную информацию. Выходная информация передается устройству вывода (например, выводу безопасности) и преобразуется в электрический сигнал. Данный сигнал соединяется с исполнительным устройством, которое выполняет физическую реакцию, например, отключение питания привода.

Основные допущения в случае коммуникационных ошибок следующие:

- все компоненты работают асинхронно;
- обработка входных(ой) сигналов/информации не зависит от обработки выходных(ой) сигналов/информации. Это означает, что каждая сторона может обладать своим поведением во времени;
- для того чтобы вычислить время реакции функции безопасности, должна допускаться только одна ошибка или один отказ во всей системе. Предполагается, что эта ошибка или отказ произошел в той части пути сигнала, которая формирует наибольшую разницу во времени между его временем

задержки в худшем случае и временем сторожевого таймера. Это означает, что одновременные отказы не рассматриваются.

В таблице 42 определены времена для компонентов.

Таблица 42 — Определение времен

Время	Название	Описание
T_SFR	Время реакции функции безопасности	Время реакции функции безопасности от физического ввода до реакции на исполнительном устройстве
T_InCon	Время соединения ввода	Время передачи сигнала физического ввода в логический узел безопасности
T_OutCon	Время соединения вывода	Время передачи расчитанного сигнала вывода от логического узла безопасности исполнительному устройству
T_S	Время датчика	Время преобразования на датчике безопасности
T_I	Время ввода	Время задержки вводного устройства безопасности
T_Com	Время коммуникаций	Время коммуникационного цикла для коммуникационной сети
T_L	Время логического узла	Время задержки логического узла (цикл)
T_O	Время вывода	Время задержки выводного устройства безопасности
T_A	Время исполнительного устройства	Время преобразования на исполнительном устройстве безопасности
T_WD_In	Время сторожевого таймера ввода	Время сторожевого таймера FSoE для соединения ввода
T_WD_Out	Время сторожевого таймера вывода	Время сторожевого таймера FSoE для соединения вывода
ΔT	запас времени сторожевого таймера	Дополнительный запас времени для минимального сторожевого таймера

Так как принято считать, что все компоненты работают асинхронно, наихудшее время для каждого компонента является удвоенным временем задержки каждого компонента. Это происходит, если «двигающаяся» информация становится доступной сразу после запуска процесса. Времена наихудших случаев помечаются суффиксом a_wc.

9.3.2 Установление времени сторожевого таймера FSoE

На рисунке 13 показана базовая схема для вычисления времени сторожевого таймера FSoE для соединения ввода и вывода.

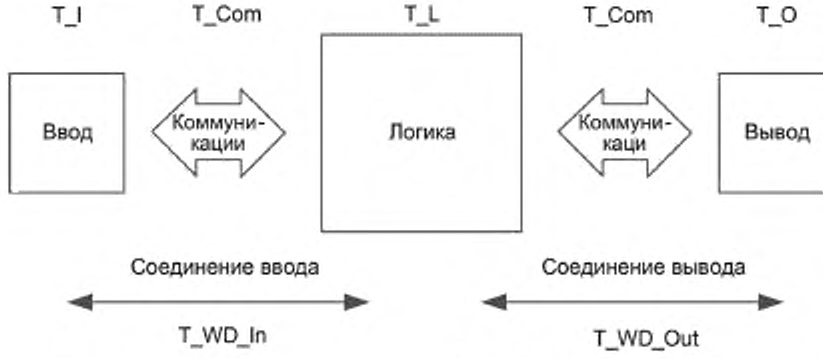


Рисунок 13 — Вычисление времени сторожевого таймера FSoE для соединений ввода и вывода

Для того чтобы определить время сторожевого таймера соединения ввода T_WD_In, можно использовать уравнение (1):

$$T_WD_In = T_I_wc + T_Com_wc + T_L_wc + T_Com_wc + \Delta T \quad (1)$$

$$= 2 \times T_I + 4 \times T_Com + 2 \times T_L + \Delta T.$$

Аналогично уравнение (2) позволяет вычислить время сторожевого таймера соединения вывода T_WD_Out:

$$T_WD_Out = T_Com_wc + T_L_wc + T_Com_wc + T_O_wc + \Delta T \quad (2)$$

$$= 4 \times T_Com + 2 \times T_L + 2 \times T_O + \Delta T.$$

9.3.3 Вычисление наихудшего времени реакции функции безопасности

На рисунке 14 показана базовая схема для вычисления наихудшего времени реакции функции безопасности.

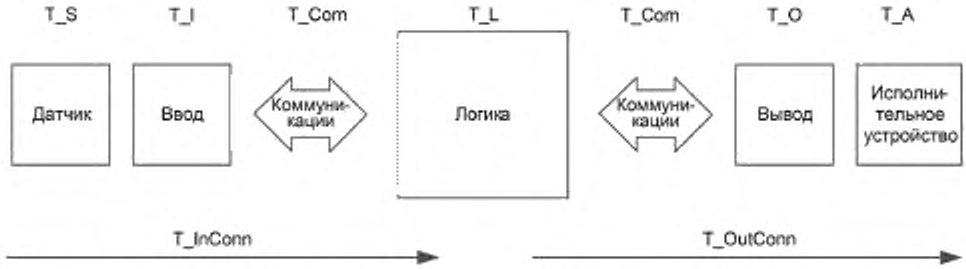


Рисунок 14 —Вычисление наихудшего времени реакции функции безопасности

Время передачи информации сигнала датчика логическому узлу безопасности, то есть время T_InConn, может быть вычислено как:

$$T_InConn = T_S_wc + T_I_wc + T_Com_wc + T_L_wc \quad (3)$$

$$= 2 \times T_S + 2 \times T_I + 2 \times T_Com + 2 \times T_L.$$

Наихудшее время получения информации безопасного состояния от сигнала датчика логическим узлом безопасности или время T_InConn_wc — это время в том случае, когда входные коммуникации прерываются и истекает время сторожевого таймера соединения ввода. В этом случае в логике безопасности используются отказоустойчивые значения входных сигналов. Это время может быть вычислено как:

$$T_InConn_wc = T_S_wc + T_WD_In \quad (4)$$

$$= 2 \times T_S + T_WD_In.$$

Время на передачу вычисленного выходного сигнала от логического узла безопасности исполнительному устройству или время T_OutConn может быть вычислено как:

$$T_InConn = T_L_wc + T_Com_wc + T_O_wc + T_A_wc \quad (5)$$

$$= 2 \times T_L + 2 \times T_Com + 2 \times T_O + 2 \times T_A.$$

Наихудшее время на передачу вычисленного выходного сигнала от логического узла безопасности исполнительному устройству или время T_OutConn_wc, это время в случае, когда выходные коммуникации прерываются и истекает время сторожевого таймера соединения вывода. В таком случае активируются отказоустойчивые значения в устройстве вывода. Это время может быть вычислено как:

$$T_OutConn_wc = T_L_wc + T_WD_Out + T_A_wc \quad (6)$$

$$= 2 \times T_L + T_WD_ Out + 2 \times T_A.$$

Для того чтобы вычислить время реакции функции безопасности, должна допускаться только одна ошибка или один отказ на том пути сигнала, который создает максимальную разницу во времени между его наихудшим временем задержки и временем сторожевого таймера. Это означает, что одновременные отказы не рассматриваются.

Для определения наихудшего времени реакции функции безопасности или времени T_SFR_wc используется уравнение (7):

$$T_SFR_wc = \max\{T_InConn_wc + T_OutConn; T_OutConn_wc + T_InConn\}. \quad (7)$$

При необходимости изготовители системы должны предоставлять индивидуальный адаптированный метод вычисления.

9.4 Длительность запросов на обслуживание

Длительность запросов на обслуживание приложения, связанного с безопасностью, коммуникационным уровнем безопасности может быть равна времени процесса безопасности или таймауту FSCP 12/1 (сторожевой таймер FSoE) или превышать их.

9.5 Ограничения на вычисление характеристик системы

9.5.1 Общие положения

FSCP 12/1 не накладывает никаких ограничений на:

- минимальное время цикла коммуникаций;
- количество данных безопасности, приходящихся на одно устройство FSoE;
- основную коммуникационную систему.

Все устройства должны быть электробезопасными, используя SELV/PELV.

Все устройства спроектированы для нормальных промышленных сред в соответствии с МЭК 61000-6-2 или МЭК 61131-2 и обеспечивают повышенную защищенность в соответствии с МЭК 61326-3-1 или МЭК 61326-3-2.

Коммуникационный путь выбирается произвольно. Он может быть системой полевых шин, Ethernet'ом или реализован другими подобными средствами, оптоволоконными кабелями, медными кабелями или даже путем беспроводной передачи. Нет никаких ограничений или требований для использования шин или других устройств в коммуникации.

Дополнительное внесение трех нулевых октетов в вычисление CRC, вместе с наследованием CRC гарантирует независимость основной коммуникации, даже если используется один CRC полином.

Коммуникационный интерфейс в устройствах безопасности может быть одноканальным интерфейсом. Такой интерфейс может быть избыточным для обеспечения готовности.

Соединение между устройствами FSoE является соединением ведущего устройства с ведомым. Ведущее устройство FSoE обладает одним или несколькими соединениями FSoE с одним или несколькими ведомыми устройствами FSoE. Ведомое устройство FSoE реагирует только на ведущее устройство FSoE. Система может различать до 65 535 соединений.

9.5.2 Ограничения, полученные в результате вероятностного анализа

Каждая обнаруженная ошибка в коммуникациях безопасности должна инициировать переход в состояние сброса, т. е. в безопасное состояние. Такой переход не должен происходить более одного раза за 5 часов, т. е. частота возникновения остаточной ошибки должна быть лучше, чем 10^{-2}/ч.

Доказано, что CRC полином с включением трех нулевых октетов (так называемых виртуальных битов) гарантирует независимость для стандартной проверки основной коммуникационной системы.

PDU Типа 12 состоит из части, связанной с безопасностью, и стандартной части. Часть безопасности встраивается в стандартную часть. На рисунке 15 показан PDU, состоящий из SafetyData ND_{safety} [1)], виртуальных битов с длиной d_{safety} = 24 бита, FCS_{safety} безопасности, стандартного $ND_{standard}$ полезных данных и стандартного $FCS_{standard}$.

---

1) Safety — безопасность.

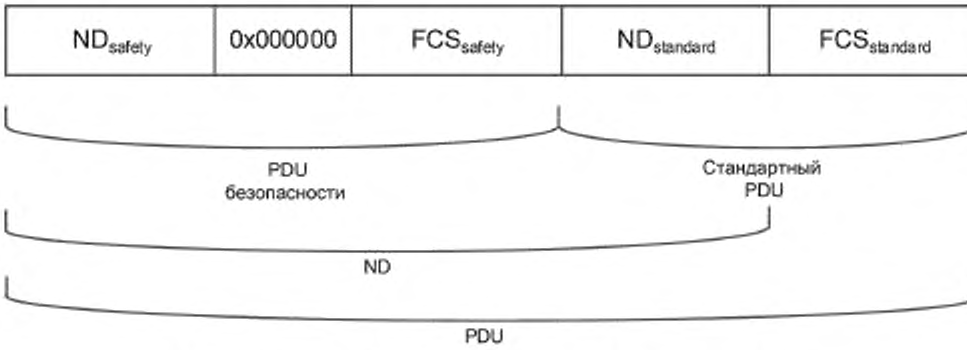


Рисунок 15 — PDU безопасности, встроенный в стандартный PDU

Были получены следующие требования:

- $x^{dsafety}+1$ и генератор полиномов основные друг для друга;
- число виртуальных битов d_{safety} меньше или равно числу битов для стандартной части ($d_{safety} \leq n_{standard}$);
- частота возникновения остаточных ошибок ниже 10^{-9}/ч.

При использовании примитивного полинома безопасности 139B7h эти требования выполняются при следующих условиях:

- число битов данных безопасности — 8 или 16 ($ND_{safety} = 8$ или $ND_{safety} = 16$);
- число виртуальных битов равно 24 ($d_{safety} = 24$);
- минимальное число стандартных битов — 16 ($ND_{standard} \geq 16$);
- максимальное число стандартных битов — 12 144 ($ND_{standard} \leq 12\ 144$).

Примечание — Было доказано, что до 12 144 битов (1 518 октетов) используется для Ethernet в качестве максимальной длины DPDU;

- стандартные биты могут снова содержать блоки данных безопасности, состоящие из данных безопасности и FCS_{safety}.

На рисунке 16 показана частота возникновения остаточных ошибок для данных безопасности из 8, 16 и 24 битов. При максимальной вероятности возникновения битовой ошибки, равной 10^{-2}/ч, вероятность возникновения остаточной ошибки ниже 10^{-9}/ч для 8- и 16-битовых данных безопасности. 24-битовые данные безопасности не используются в рамках данного протокола.

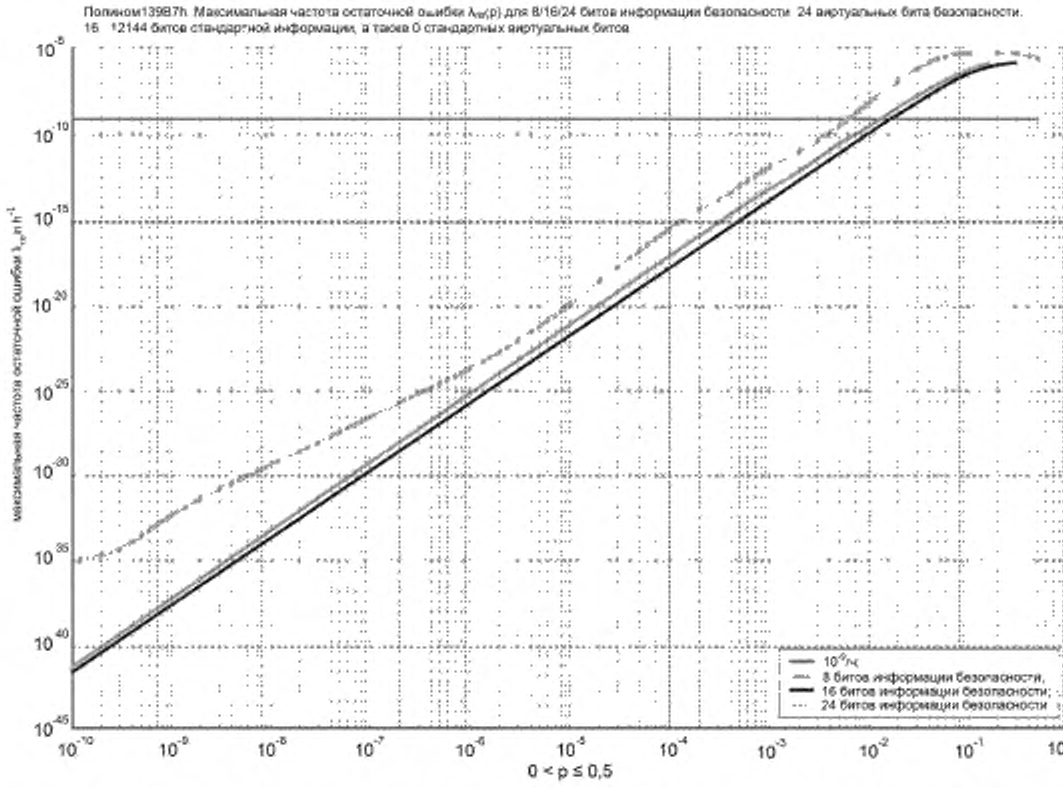


Рисунок 16 — Частота возникновения остаточной ошибки для 8/16/24-битовых данных безопасности и стандартных данных до 12 144 битов

9.6 Техническое обслуживание

Этот протокол не содержит специальных требований к техническому обслуживанию.

9.7 Руководство по безопасности

Специалисты по внедрению настоящего стандарта должны предоставить руководство по безопасности, содержащее, как минимум, следующую информацию:

- руководство по безопасности должно оповещать пользователей об ограничениях на вычисления системных характеристик, см. 9.5;
- руководство по безопасности должно оповещать пользователей об ответственностях, связанных с надлежащей параметризацией устройства.

Кроме требований данного раздела, руководство по безопасности должно удовлетворять всем требованиям МЭК 61508.

10 Оценка

Настоятельно рекомендуется, чтобы специалисты по внедрению FSCP 12/1 получили результаты верификации всех аспектов функциональной безопасности изделия от независимого компетентного органа оценки, включая протокол и все приложения. Настоятельно рекомендуется, чтобы специалисты по внедрению FSCP 12/1 получили доказательство того, что независимым компетентным органом оценки был выполнен подходящий тест на соответствие.

Приложение А
(справочное)

Дополнительная информация для профилей коммуникаций функциональной безопасности CPF 12

А.1 Вычисление хэш функции

Нижеприведенный код для PDU безопасности представляет из себя пример того, как вычисляются CRC коды PDU безопасности. В таблицах учитываются три нулевых младших разряда.

```
*************************************************************
** Параметр: psPacket - FSCP12/1 Safety PDU
**           startCrc - Начальное значение вычисления CRC
**           seqNo    - SeqNo
**           oldCRC   - CRC_0 последнего принятого/отправленного PDU ведомого
**                    - устройства безопасности
**           bRcvDir  - bRcvDir = True: вычисления CRC принятого кадра
**                      bRcvDir = False: вычисления CRC для отправленного кадра
**           size     - размер PDU безопасности
**
** Возвращает: bSuccess - TRUE: CRC верен
**
*************************************************************/
UINT8 CalcCrc(SAFETY_PDU *psPacket, UINT16 startCrc, UINT16 * seqNo, UINT16 oldCrc,
UINT8 bRcvDir, UINT8 size)
{
UINT16 w1,w2;            // временные значения
UINT16 crc;
UINT16 crc_common;       // общая часть вычисления CRC,
                         // включает CRC_0, Conn-ID, Sequence-No., Cmd
UINT8 *pCrc = &psPacket->au8Data[2]; // указатель на младший байт CRC
UINT8 *pSafeData                     // указатель на младший байт SafeData

if ( size > 6 )                  // т.е. 2 или кратное двум данным безопасности
     pCrc++;                     //-> младший байт Crc0 при байтовом
                                 // смещении 3 вместо 2
do
{
     crc = 0;                    // сброс crc

// Последовательность для вычисления:
// старый CRC-Lo, старый CRC-Hi, ConnId-Lo, ConnId-Hi, SeqNo-Lo, SeqNo-Hi, Command,
// (Индекс,) Данные

// CRC-Lo
w1 = aCRCTab1[(UINT8 *) &crc)[HI_BYTE]];      // Обратитесь к CRC-таблице
w2 = aCRCTab2[(UINT8 *) &startCrc)[0]];       // Обратитесь к CRC-таблице
w1 = w1 XOR w2;
((UINT8 *) &crc)[HI_BYTE]  = ((UINT8 *) &w1)[HI_BYTE] XOR ((UINT8 *)
                             &crc)[LO_BYTE];
((UINT8 *) &crc)[LO_BYTE]  = ((UINT8 *) &w1)[LO_BYTE] ;

// CRC-Hi
w1 = aCRCTab1[(UINT8 *) &crc)[HI_BYTE]];
w2 = aCRCTab2[(UINT8 *) &startCrc)[1]] ;
w1 = w1 XOR w2;
((UINT8 *) &crc)[HI_BYTE] = ((UINT8 *) &w1)[HI_BYTE]  XOR ((UINT8 *)
                             &crc)[LO_BYTE];
((UINT8 *) &crc)[LO_BYTE]  = ((UINT8 *) &w1)[LO_BYTE] ;
```

```
// ConnId-Lo
w1 = aCRCTab1[((UINT8 *) &crc)[HI_BYTE]];
w2 = aCRCTab2[psPacket->au8Data[size-2]];
w1 = w1 XOR w2;
((UINT8 *) &crc)[HI_BYTE] = ((UINT8 *) &w1)[HI_BYTE] XOR ((UINT8 *)
                            &crc)[LO_BYTE];
((UINT8 *) &crc)[LO_BYTE] = ((UINT8 *) &w1)[LO_BYTE];

// ConnId-Hi
w1 = aCRCTab1[((UINT8 *) &crc)[HI_BYTE]];
w2 = aCRCTab2[psPacket->au8Data[size-1]];
w1 = w1 XOR w2;
((UINT8 *) &crc)[HI_BYTE] = ((UINT8 *) &w1)[HI_BYTE] XOR ((UINT8 *)
                            &crc)[LO_BYTE];
((UINT8 *) &crc)[LO_BYTE] = ((UINT8 *) &w1)[LO_BYTE];

// SeqNo-Lo
w1 = aCRCTab1[((UINT8 *) &crc)[HI_BYTE]];
w2 = aCRCTab2[((UINT8 *) seqNo)[LO_BYTE]];
w1 = w1 XOR w2;
((UINT8 *) &crc)[HI_BYTE] = ((UINT8 *) &w1)[HI_BYTE] XOR ((UINT8 *)
                            &crc)[LO_BYTE];
((UINT8 *) &crc)[LO_BYTE] = ((UINT8 *) &w1)[LO_BYTE];

// SeqNo-Hi
w1 = aCRCTab1[((UINT8 *) &crc)[HI_BYTE]];
w2 = aCRCTab2[((UINT8 *) seqNo)[HI_BYTE]];
w1 = w1 XOR w2;
((UINT8 *) &crc)[HI_BYTE] = ((UINT8 *) &w1)[HI_BYTE] XOR ((UINT8 *)
                            &crc)[LO_BYTE];
((UINT8 *) &crc)[LO_BYTE] = ((UINT8 *) &w1)[LO_BYTE];

// Команда
w1 = aCRCTab1[((UINT8 *) &crc)[HI_BYTE]];
w2 = aCRCTab2[psPacket->au8Data[OFFS_COMMAND]];
w1 = w1 XOR w2;
((UINT8 *) &crc)[HI_BYTE] = ((UINT8 *) &w1)[HI_BYTE] XOR ((UINT8 *)
                           &crc)[LO_BYTE];
((UINT8 *) &crc)[LO_BYTE] = ((UINT8 *) &w1)[LO_BYTE];

// часть CRC общая для всех других crc-вычислений сохранена
crc_common = crc;

// Данные [0]
w1 = aCRCTab1[((UINT8 *) &crc)[HI_BYTE]];
w2 = aCRCTab2[psPacket->au8Data[OFFS_DATA]];
w1 = w1 XOR w2;
((UINT8 *) &crc)[HI_BYTE] = ((UINT8 *) &w1)[HI_BYTE] XOR ((UINT8 *)
                           &crc)[LO_BYTE];
((UINT8 *) &crc)[LO_BYTE] = ((UINT8 *) &w1)[LO_BYTE];
// если 2 байта данных безопасности - вычислить следующий байт crc
if ( size > 6 )
{
// Данные [1]
w1 = aCRCTab1[((UINT8 *) &crc)[HI_BYTE]];
w2 = aCRCTab2[psPacket->au8Data[OFFS_DATA+1]];
w1 = w1 XOR w2;
((UINT8 *) &crc)[HI_BYTE] = ((UINT8 *) &w1)[HI_BYTE] XOR ((UINT8 *)
                            &crc)[LO_BYTE];
((UINT8 *) &crc)[LO_BYTE] = ((UINT8 *) &w1)[LO_BYTE];
}
```

```
// UPDATE_SEQ_NO
seqNo[0]++;
if (seqNo[0] == 0)
seqNo[0]++;

} while ( crc == oldCrc && (bRcvDir & NEW_CRC) != 0 );
// до тех пор пока результирующий crc такой же, как и oldCrc

if (bRcvDir) // для направления приёма
{     if ( ((UINT8 *) &crc)[HI_BYTE] == pCrc[OFFS_CRC_HI-OFFS_CRC_LO]
        && ((UINT8 *) &crc)[LO_BYTE] == pCrc[0] )
{      // для направления приема
       // CRC верен
       bSuccess = TRUE;
}
}
else                  // для направления передачи
{
// введите контрольную сумму Checksum
pCrc[OFFS_CRC_HI-OFFS_CRC_LO] = ((UINT8 *) &crc)[HI_BYTE];
pCrc[0] = ((UINT8 *) &crc)[LO_BYTE];
}

// если передано больше, чем 2 байта данных безопасности,
// CRC_1 и остальное должно быть вычислено
if ( size > 10 )
{
UINT16 w;      = 1;
pSafeData     = pCrc+2;    // установить pSafeData в младший байт SafeData
                           // следующей части = SafeData[2].
pCrc +        = 4;         // установить pCrc в младший байт CRC_2
size -= 7;                 // вычесть первую часть кадра
while ( size >= 4 )        // пока другие части следуют
{
// Start-CRC
crc = crc_common;      // данная часть, уже вычислена выше

// 2 (Bit 0-7)            // вычислить индекс
w1 = aCRCTab1[((UINT8 *) &crc)[HI_BYTE]];
w2 = aCRCTab2[((UINT8 *) &i)[LO_BYTE]];
w1 = w1 XOR w2;
((UINT8 *) &crc)[HI_BYTE] = ((UINT8 *) &w1)[HI_BYTE] XOR ((UINT8 *)
                            &crc)[LO_BYTE];
((UINT8 *) &crc)[LO_BYTE] = ((UINT8 *) &w1)[LO_BYTE];

// 2 (Bit 8-15)
w1 = aCRCTab1[((UINT8 *) &crc)[HI_BYTE]];
w2 = aCRCTab2[((UINT8 *) &i)[HI_BYTE]];
w1 = w1 XOR w2;
((UINT8 *) &crc)[HI_BYTE] = ((UINT8 *) &w1)[HI_BYTE] XOR ((UINT8 *)
                             &crc)[LO_BYTE];
((UINT8 *) &crc)[LO_BYTE] = ((UINT8 *) &w1)[LO_BYTE];

// Данные 2*2
w1 = aCRCTab1[((UINT8 *) &crc)[HI_BYTE]];
w2 = aCRCTab2[pSafeData[0]];
w1 = w1 XOR w2;
((UINT8 *) &crc)[HI_BYTE] = ((UINT8 *) &w1)[HI_BYTE] XOR ((UINT8 *)
                              &crc)[LO_BYTE];
((UINT8 *) &crc)[LO_BYTE] = ((UINT8 *) &w1)[LO_BYTE];
```

```
// Данные 2*i+1
w1 = aCRCTab1[((UINT8 *) &crc)[HI_BYTE]];
w2 = aCRCTab2[pSafeData[1]];
w1 = w1 XOR w2;
((UINT8 *) &crc)[HI_BYTE] = ((UINT8 *) &w1)[HI_BYTE] XOR ((UINT8 *)
                            &crc)[LO_BYTE];
((UINT8 *) &crc)[LO_BYTE] = ((UINT8 *) &w1)[LO_BYTE];

if ( ((UINT8 *) &crc)[HI_BYTE] == pCrc [1]
  && ((UINT8 *) &crc)[LO_BYTE] == pCrc [0] )
{
        // CRC верен
}
else
{
bSuccess = FALSE;
if ( bRcvDir == 0)    // для направления передачи
{
       // введите контрольную сумму Checksum
pCrc [1] = ((UINT8 *) &crc)[HI_BYTE];
pCrc [0] = ((UINT8 *) &crc)[LO_BYTE];
}
}

size -= 4;              // вычесть данную часть кадра
pSafeData += 4;         // установить в следующий младший байт SafeData
pCrc0 += 4;             // установить в следующий младший байт CRC_i
i++;                    // индекс приращения
}
}

return bSuccess;
}
```

Используются следующие две таблицы:

```
aCrcTab1: ARRAY[0..255] OF WORD :=
16#0000,16#39B7,16#736E,16#4AD9,16#E6DC,16#DF6B,16#95B2,16#AC05,16#F40F,16#CDB8,
16#8761,16#BED6,16#12D3,16#2B64,16#61BD,16#580A,16#D1A9,16#E81E,16#A2C7,16#9B70,
16#3775,16#0EC2,16#441B,16#7DAC,16#25A6,16#1C11,16#56C8,16#6F7F,16#C37A,16#FACD,
16#B014,16#89A3,16#9AE5,16#A352,16#E98B,16#D03C,16#7C39,16#458E,16#0F57,16#36E0,
16#6EEA,16#575D,16#1D84,16#2433,16#8836,16#B181,16#F358,16#C2EF,16#1B4C,16#72FB,
16#3822,16#0195,16#AD90,16#9427,16#DEFE,16#E749,16#3F43,16#86F4,16#CC2D,16#F59A,
16#599F,16#6028,16#2AF1,16#1346,16#0C7D,16#35CA,16#7F13,16#46A4,16#EAA1,16#D316,
16#99CF,16#A078,16#F872,16#C1C5,16#8B1C,16#B2AB,16#1EAE,16#2719,16#6DC0,16#5477,
16#DDD4,16#E463,16#AEBA,16#970D,16#3308,16#023F,16#4866,16#7ID1,16#29D3,16#106C,
16#5AB5,16#6302,16#CF07,16#F6B0,16#BC69,16#85DE,16#9698,16#AF2F,16#E5F6,16#DC41,
16#7044,16#49F3,16#032A,16#3A9D,16#6291,16#5B20,16#11F9,16#284E,16#844B,16#BDFC,
16#F725,16#CE92,16#4731,16#7E86,16#345F,16#0DE8,16#A1ED,16#985A,16#D283,16#EB34,
16#B33E,16#8A89,16#C050,16#F9E7,16#55E2,16#6C55,16#268C,16#1F3B,16#18FA,16#214D,
16#6B94,16#5223,16#FE26,16#C791,16#8D48,16#B4FF,16#ECF5,16#D542,16#9F93,16#A62C,
16#0A29,16#339E,16#7947,16#40F0,16#C953,16#F0E4,16#BA3D,16#838A,16#2F8F,16#1638,
16#5CE1,16#6556,16#3D5C,16#04EB,16#4E32,16#7785,16#DB80,16#E237,16#A8EE,16#9159,
16#821F,16#BBA8,16#F171,16#C8C6,16#64C3,16#5D74,16#17AD,16#2E1A,16#7610,16#4FA7,
16#057E,16#3CC9,16#90CC,16#A97B,16#E3A2,16#DA15,16#53B6,16#6A01,16#20D8,16#196F,
16#B56A,16#8CDD,16#C604,16#FFB3,16#A7B9,16#9E0E,16#D4D7,16#ED60,16#4165,16#78D2,
16#3203,16#0BBC,16#1487,16#2D30,16#67E9,16#5E5E,16#F25B,16#CBEC,16#8135,16#B882,
16#E088,16#D93F,16#93E6,16#AA51,16#0654,16#3FE3,16#753A,16#4C8D,16#C52E,16#FC99,
16#B640,16#8FF7,16#23F2,16#1A45,16#509C,16#692B,16#3121,16#0896,16#424F,16#7BF8,
16#D7FD,16#EE4A,16#A493,16#9D24,16#8E62,16#B7D5,16#FD0C,16#C4B3,16#68BE,16#5109,
16#1BD0,16#2267,16#7A6D,16#43DA,16#0903,16#30B4,16#9C31,16#A506,16#EF0F,16#D668,
16#5FCB,16#667C,16#2CA5,16#1512,16#3917,16#80A0,16#CA79,16#F3CE,16#ABC4,16#9273,
16#D8AA,16#E11D,16#4D18,16#74AF,16#3E76,16#07C1;
```

```
aCrcTab2: ARRAY[0..255] OF WORD :=
16#0000,16#7648,16#EC90,16#9AD8,16#E097,16#96DF,16#0C07,16#7A4F,16#F899,16#8ED1,
16#1409,16#6241,16#F80E,16#6E46,16#F49E,16#82D6,16#C885,16#BECD,16#2415,16#525D,
16#2812,16#5E5A,16#C482,16#B2CA,16#3D1C,16#4654,16#DC8C,16#AAC4,16#D08B,16#A6C3,
16#3C1B,16#4A53,16#A8BD,16#DEF5,16#442D,16#3265,16#482A,16#3E62,16#A4BA,16#D2F2,
16#5024,16#266C,16#BCB4,16#CAFC,16#B0B3,16#C6FB,16#5C23,16#2A6B,16#6038,16#1670,
16#8CA8,16#FAE0,16#80AF,16#F6E7,16#6C3F,16#1A77,16#98A1,16#EEE9,16#7431,16#0279,
16#7836,16#0E7E,16#94A6,16#E2EE,16#68CD,16#1E85,16#845D,16#F215,16#885A,16#FE12,
16#64CA,16#1282,16#9054,16#E61C,16#7CC4,16#0A8C,16#70C3,16#068B,16#9C53,16#EA1B,
16#A048,16#D600,16#4CD8,16#3A90,16#40DF,16#3697,16#AC4F,16#DA07,16#58D1,16#2E99,
16#B441,16#C209,16#B846,16#CE0E,16#54D6,16#229E,16#C070,16#B638,16#2CE0,16#5AA8,
16#20E7,16#56AF,16#CC77,16#BA3F,16#38E9,16#4EA1,16#D479,16#A231,16#D87E,16#AE36,
16#34EE,16#42A6,16#08F5,16#7EBD,16#E465,16#922D,16#E862,16#9E2A,16#04F2,16#72BA,
16#F06C,16#8624,16#1CFC,16#6AB4,16#10FB,16#66B3,16#FC6B,16#8A23,16#D19A,16#A7D2,
16#3D0A,16#4B42,16#310D,16#4745,16#DD9D,16#ABD5,16#2903,16#5F4B,16#C593,16#B3DB,
16#C994,16#BFDC,16#2504,16#534C,16#291F,16#6F57,16#F58F,16#83C7,16#F988,16#8FC0,
16#1518,16#6350,16#E186,16#97CE,16#0D16,16#7B5E,16#0111,16#7759,16#ED81,16#9BC9,
16#7927,16#0F6F,16#95B7,16#E3FF,16#99B0,16#EFF8,16#7520,16#0368,16#81BE,16#F7F6,
16#6D2E,16#1B66,16#6129,16#1761,16#8DB9,16#FBF1,16#B1A2,16#C7EA,16#5D32,16#2B7A,
16#5135,16#277D,16#BDA5,16#CBED,16#493B,16#3F73,16#A5AB,16#D3E3,16#A9AC,16#DFE4,
16#453C,16#3374,16#B957,16#CF1F,16#55C7,16#238F,16#59C0,16#2F88,16#B550,16#C318,
16#41CE,16#3786,16#AD5E,16#DB16,16#A159,16#D711,16#4DC9,16#3B81,16#71D2,16#079A,
16#9D42,16#EB0A,16#9145,16#E70D,16#7DD5,16#0B9D,16#894B,16#FF03,16#65DB,16#1393,
16#69DC,16#1F94,16#854C,16#F304,16#11EA,16#67A2,16#FD7A,16#8B32,16#F17D,16#8735,
16#1DED,16#6BA5,16#E973,16#9F3B,16#05E3,16#73AB,16#09E4,16#7FAC,16#E574,16#933C,
16#D96F,16#AF27,16#35FF,16#43B7,16#39F8,16#4FB0,16#D568,16#A320,16#21F6,16#57BE,
16#CD66,16#BB2E,16#C161,16#B729,16#2DF1,16#5BB9;
```

Приложение В
(справочное)

Информация для оценки профилей коммуникаций функциональной безопасности CPF 12

Информация о тестовых лабораториях, в которых испытывают и подтверждают соответствие изделий FSCP 8/1 стандарту МЭК 61784-3-12, может быть получена в национальных комитетах МЭК или в следующих организациях:

EtherCAT Technology Group
Ostendstrasse 196
90482 Nuremberg
GERMANY
Телефон: +49-911-54056-20
Факс: +49-911-54056-29
E-mail: gost.e-ecolog.ru
URL: www.ethercat.org

Приложение ДА
(справочное)

Сведения о соответствии ссылочных международных стандартов национальным стандартам

Таблица ДА.1

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование соответствующего национального стандарта
IEC 60204-1	IDT	ГОСТ Р МЭК 60204-1—2007 «Безопасность машин. Электрооборудование машин и механизмов. Часть 1. Общие требования»
IEC 61000-6-2	MOD	ГОСТ Р 51317.6.2—2007 (МЭК 61000-6-2—2005) «Совместимость технических средств электромагнитная. Устойчивость к электромагнитным помехам технических средств, применяемых в промышленных зонах. Требования и методы испытаний»
IEC 61131-2	IDT	ГОСТ IEC 61131-2—2012 «Контроллеры программируемые. Часть 2. Требования к оборудованию и испытания»
IEC 61158-2	—	*
IEC 61158-3-12	—	*
IEC 61158-4-12	—	*
IEC 61158-5-12	—	*
IEC 61158-5-10	—	*
IEC 61158-6-12	—	*
IEC 61326-3-1	—	*
IEC 61326-3-2	—	*
IEC 61508 (все части)	IDT	ГОСТ Р МЭК 61508—2012 (все части). «Функциональная безопасность систем электрических, электронных, программируемых электронных, связанных с безопасностью»
IEC 61784-2	—	*
IEC 61784-3	—	*
IEC 61918	—	*

* Соответствующий национальный стандарт отсутствует. До его утверждения рекомендуется использовать перевод на русский язык данного международного стандарта.

Примечание — В настоящей таблице использованы следующие условные обозначения степени соответствия стандартов:

- IDT — идентичные стандарты;
- MOD — модифицированный стандарт.

Библиография

[1] IEC 60050 (all parts), International Electrotechnical Vocabulary

Примечание — См. также IEC Multilingual Dictionary — Electricity, Electronics and Telecommunications (доступен на CD-ROM и по адресу <gost.e-ecolog.ru >).

[2] IEC/TS 61000-1-2, Electromagnetic compatibility (EMC) — Part 1-2: General —Methodology for the achievement of the functional safety of electrical and electronic equipment with regard to electromagnetic phenomena

[3] IEC 61131-6, Programmable controllers — Part 6: Functional safety

[4] IEC 61158 (all parts), Industrial communication networks — Fieldbus specifications

[5] IEC 61496 (all parts), Safety of machinery — Electro-sensitive protective equipment

[6] IEC 61508-1:2010, Functional safety of electrical/electronic/programmable electronic safety-related systems — Part 1: General requirements

[7] IEC 61508-4:2010, Functional safety of electrical/electronic/programmable electronic safety-related systems — Part 4: Definitions and abbreviations

[8] IEC 61508-5:2010, Functional safety of electrical/electronic/programmable electronic safety-related systems — Part 5: Examples of methods for the determination of safety integrity levels

[9] IEC 61511 (all parts), Functional safety — Safety instrumented systems for the process industry sector

[10] IEC 61784-1, Industrial communication networks — Profiles — Part 1: Fieldbus profiles

[11] IEC 61784-4, Industrial communication networks — Profiles — Part 4: Secure communications for fieldbuses

[12] IEC 61784-5 (all parts), Industrial communication networks — Profiles — Part 5: Installation of fieldbuses — Installation profiles for CPF x

[13] IEC 61800-5-2, Adjustable speed electrical power drive systems — Part 5-2: Safety requirements — Functional

[14] IEC/TR 62059-11, Electricity metering equipment — Dependability — Part 11: General concepts

[15] IEC 62061, Safety of machinery — Functional safety of safety-related electrical, electronic and programmable electronic control systems

[16] IEC/TR 62210, Power system control and associated communications — Data and communication security

[17] IEC 62280-1, Railway applications — Communication, signalling and processing systems — Part 1: Safety-related communication in closed transmission systems

[18] IEC 62280-2, Railway applications — Communication, signalling and processing systems — Part 2: Safety-related communication in open transmission systems

[19] IEC 62443 (all parts), Industrial communication networks — Network and system security

[20] ISO/IEC Guide 51:1999, Safety aspects — Guidelines for their inclusion in standards

[21] ISO/IEC 2382-14, Information technology — Vocabulary — Part 14: Reliability, maintainability and availability

[22] ISO/IEC 2382-16, Information technology — Vocabulary — Part 16: Information theory

[23] ISO/IEC 7498 (all parts), Information technology — Open Systems Interconnection — Basic Reference Model

[24] ISO/IEC 19501, Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2

[25] ISO 10218-1, Robots for industrial environments — Safety requirements — Part 1: Robot

[26] ISO 12100-1, Safety of machinery — Basic concepts, general principles for design — Part 1: Basic terminology, methodology

[27] ISO 13849-1, Safety of machinery — Safety-related parts of control systems — Part 1: General principles for design

[28] ISO 13849-2, Safety of machinery — Safety-related parts of control systems — Part 2: Validation

[29] ISO 14121, Safety of machinery — Principles of risk assessment

[30] EN 954-1:1996, Safety of machinery — Safety related parts of control systems — General principles for design

[31] ANSI/ISA-84.00.01-2004 (all parts), Functional Safety: Safety Instrumented Systems for the Process Industry Sector

[32] VDI/VDE 2180 (all parts), Safeguarding of industrial process plants by means of process control engineering

[33] GS-ET-26, Grundsatz für die Prüfung und Zertifizierung von Bussystemen für die Übertragung sicherheitsrelevanter Nachrichten, May 2002. HVBG, Gustav-Heinemann-Ufer 130, D-50968 Köln ("Principles for Test and Certification of Bus Systems for Safety relevant Communication")

[34] ANDREW S. TANENBAUM, Computer Networks, 4th Edition, Prentice Hall, N.J., ISBN-10:0130661023, ISBN-13: 978-0130661029

[35] W. WESLEY PETERSON, Error-Correcting Codes, 2nd Edition 1981, MIT-Press, ISBN 0-262-16-039-0

[36] BRUCE P. DOUGLASS, Doing Hard Time, 1999, Addison-Wesley, ISBN 0-201-49837-5

[37] New concepts for safety-related bus systems, 3rd International Symposium "Programmable Electronic Systems in Safety Related Applications", May 1998, from Dr. Michael Schäfer, BG-Institute for Occupational Safety and Health.

[38] DIETER CONRADS, Datenkommunikation, 3rd Edition 1996, Vieweg, ISBN 3-528-245891

[39] German IEC subgroup DKE AK 767.0.4: EMC and Functional Safety, Spring 2002

[40] NFPA79 (2002), Electrical Standard for Industrial Machinery

[41] GUY E. CASTAGNOLI, On the Minimum Distance of Long Cyclic Codes and Cyclic Redundancy-Check Codes, 1989, Dissertation No. 8979 of ETH Zurich, Switzerland

[42] GUY E. CASTAGNOLI, STEFAN BRÄUER, and MARTIN HERRMANN, Optimization of Cyclic Redundancy-Check Codes with 24 and 32 Parity Bits, June 1993, IEEE Transactions On Communications, Volume 41, No. 6

[43] SCHILLER F and MATTES T: An Efficient Method to Evaluate CRC-Polynomials for Safety-Critical Industrial Communication, Journal of Applied Computer Science, Vol. 14, No 1, pp. 57-80, Technical University Press, Łódź,Poland, 2006

[44] SCHILLER F and MATTES T: Analysis of CRC-polynomials for Safety-critical Communication by Deterministic and Stochastic Automata, 6th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes, SAFEPROCESS 2006, pp. 1003-1008, Beijing, China, 2006

УДК 62-783:614.8:331.454:006.354 ОКС 13.110 Т51

Ключевые слова: промышленные сети, профили, функциональная безопасность полевых шин, спецификации для CPF 12

Редактор *А.Н. Рубин*
Технический редактор *В.Н. Прусакова*
Корректор *М.В. Бучная*
Компьютерная верстка *Е.О. Асташина*

Сдано в набор 22.12.2016. Подписано в печать 16.01.2017. Формат 60×84 1/8. Гарнитура Ариал.
Усл. печ. л. 10,70. Уч.-изд. л. 9,68. Тираж 27 экз. Зак. 76.
Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

Издано и отпечатано во ФГУП «СТАНДАРТИНФОРМ» 123995 Москва, Гранатный пер., 4
www.gostinfo.ru gost.e-ecolog.ru

gost.e-ecolog.ru