© CAN in Automation e. V.

CANopen

Application Layer and Communication Profile

CiA Draft Standard 301

Version 4.02
Date: 13 February 2002

Русская версия

Перевод
Соловьёв А.А.
Dr.D

2007

/Предисловие переводчика.

В России на сегодняшний день нет никаких приемлемых или хотя бы полных переводов данного документа. Тем не менее технология CANopen победно шагает, если не всей планете то по Европе точно, и не хотелось бы чтобы Росские разработчики плелись в хвосте.

Данный документ является ядром целой кипы документов – описывая основные понятия и механизмы CANopen. Подробные принципы построеня приборов конкретных предназначений приводятся в описаниях профилей устройств, - скачать можно попробовать отсюда http://www.can-cia.org. – требуется зарегистрироваться не стесняйтесь, денег за это они не берут.

Документ не подвергался никаким более – менее строгим редакциям. Но так как прошёл уже почти год а времени на это я всё не нахожу, решил опубликовать его в таком виде, поскольку это лучше чем ничего.

5. Определения и аббревиатуры.

CiA: международная организация CAN in Automation - "CAN в автоматизации".

COBID: идентификатор коммуникационного объекта.

PDO: объект данных процесса; обеспечивает обмен компактными данными в режиме жесткого реального времени.

RTR: удаленный запрос объекта.

SDO: сервисный объект данных; обеспечивает обмен большими объемами данных в режиме мягкого реального времени..

6. Общие положения

Для описания CAN сетей используются описанные ниже модель OSI, обьектная модель и модель взаимодействия.

6.1 Модель OSI

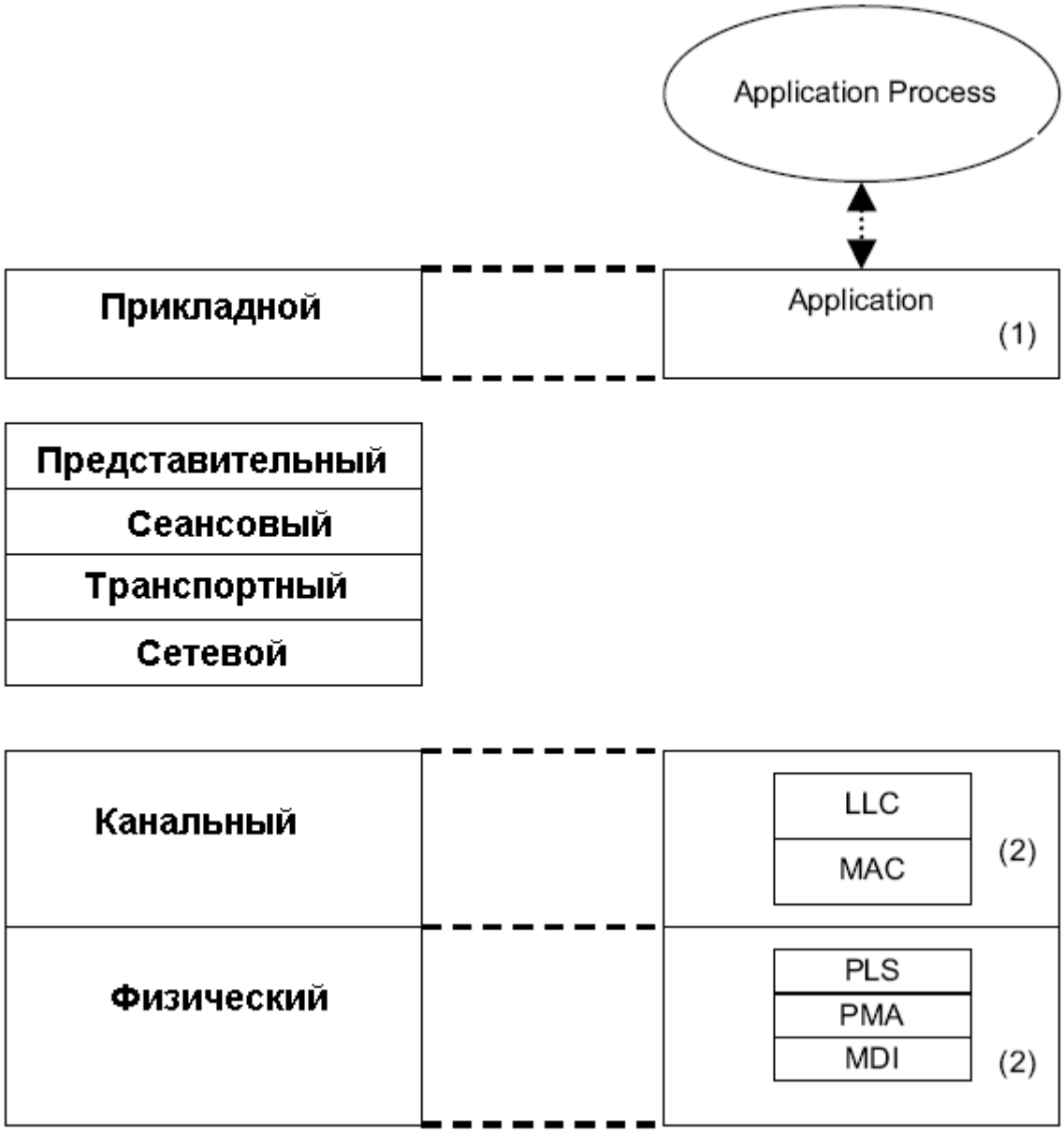


Рисунок 1. Эталонная модель *1 – спецификация приведена в этом документе
*2 – спецификация в ISO 11898

Общая идея взаимодействия может быть описана в терминах эталонной модели ISO-OSI (сама модель на левой части рисунка).

Уровень приложений: уровень приложений включает в себя концепции о конфигурации и связи в режиме реального времени и, с другой стороны, механизм синхронизации между устройствами. Функциональные возможности предоставляемые уровнем приложений CAN Open, можно разделить на несколько различных групп. Сервисные обьекты этого уровня предлагают широкий набор возможностей и все связанные с этим процессы-сервисы. Все предоставляемые услуги описываются в Описании Севисов(Service

Specification) для данного сервисного обьекта. Приложения общаются путём активизации сервисов каждого сервисного обьекта уровня приложений. Для реализации этих возможностей, данные сервисные обьекты общаются через физический уровень эталонной модели ISO-OSI.(CANNetwork) по протоколу описывающемуся в Описании протокола(Protocol Specification) этого сервисного обьекта.

Сервисные примитивы:

Сервисные примитивы есть способ взаимодействия приложений и прикладного уровня модели OSI. Существует 4 таких примитива.

- исходящий из приложения запрос сервиса(request)
- исходящая из прикладного уровня, индикация, о произошедшем внутреннем событии или о запросе сервиса(indication)
- исходящий из приложения ответ на предшествующую этому индикацию приложению(response).
- исходящее из прикладного уровня подтверждение, о том что получен ответ на ранее осуществлённый запрос сервиса(comfirm).

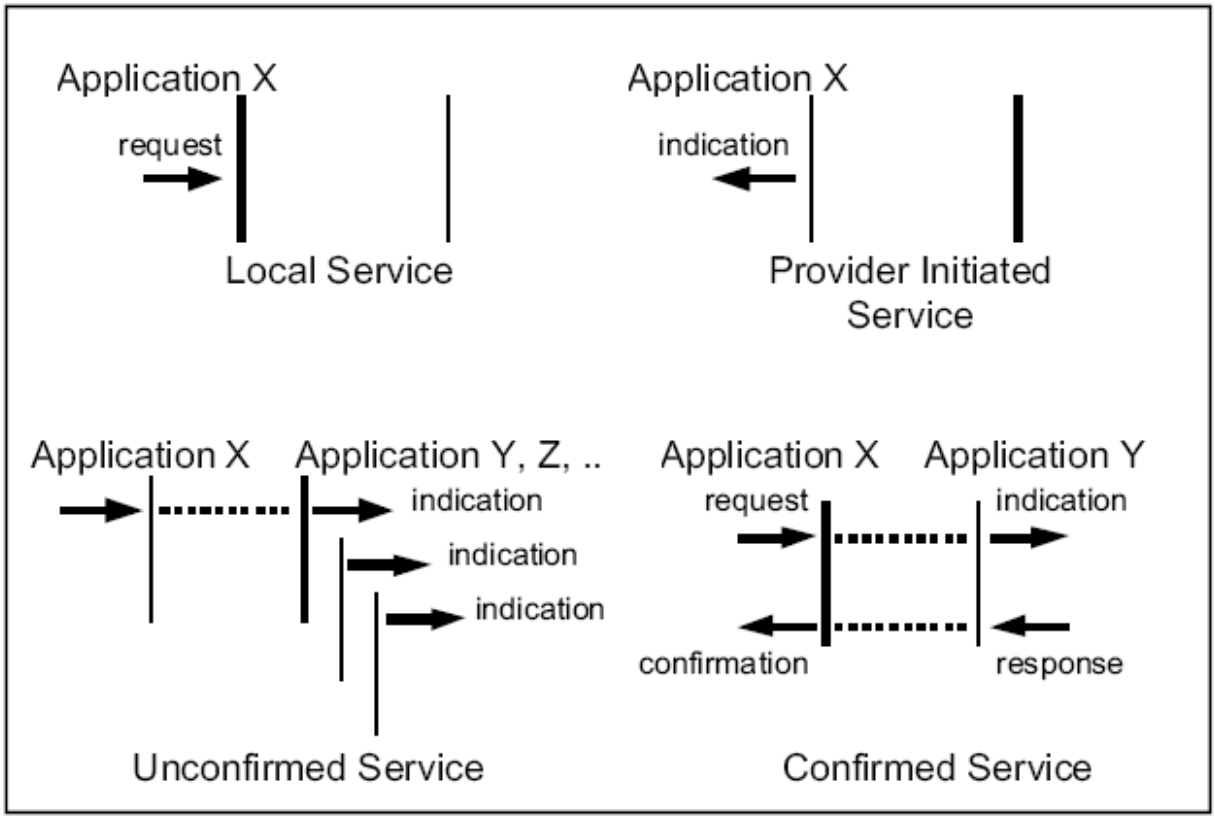


Рисунок 2. Типы сервисов

Сервисные типы определяются сервисными примитивами, через которые происходит обмен между прикладным уровнем и всеми участвующими в обмене сервисами.

- Локальный сервис(Local Service) включает только локальный сервисный обьект. Приложение посылает запрос локальному сервисному обьекту, который обрабатывается этим сервисным обьектом без обмена с удалёнными обьектами.
- Неподтверждаемый сервис(Unconfirmed Service) включает один или более равных сервисных обьектов обьектов. Приложение инициирует запрос к своему

сервисному обьекту, который передаёт этот запрос всем удалённым сервисным обьектам которые, в свою очередь передадут его своим приложениям в виде индикации. Ответ от приложений не ожидается.

- Подтверждаемый сервис(Confirmed Service) включает только один удалённый сервисный обьект. Приложение посылает запрос своему локальному сервисному обьекту. Этот запрос посылается удалённому сервисному обьекту, который передаёт его второму приложению в виде индикации. Второе приложение инициирует ответ, который передаётся первому сервисному обьекту, который уже передаёт своему приложению(запросившему) как подтверждение.
- Сервис предоставленный провайдером сервисов(Provider Initiated Service). Включает только локальный сервисный обьект, являющийся поставщиком сервиса, обнаруживает требуемое событие без предварительного запроса со стороны приложения. При обнаружении события в приложение передаётся индикация.

Неподтверждаемый и подтверждаемый сервисы в целом называются удалённые сервисы.

6.2 Обьектная модель.

6.2.1. Общее

С точки зрения обьектноц модели устройство делится как это проказано ниже.

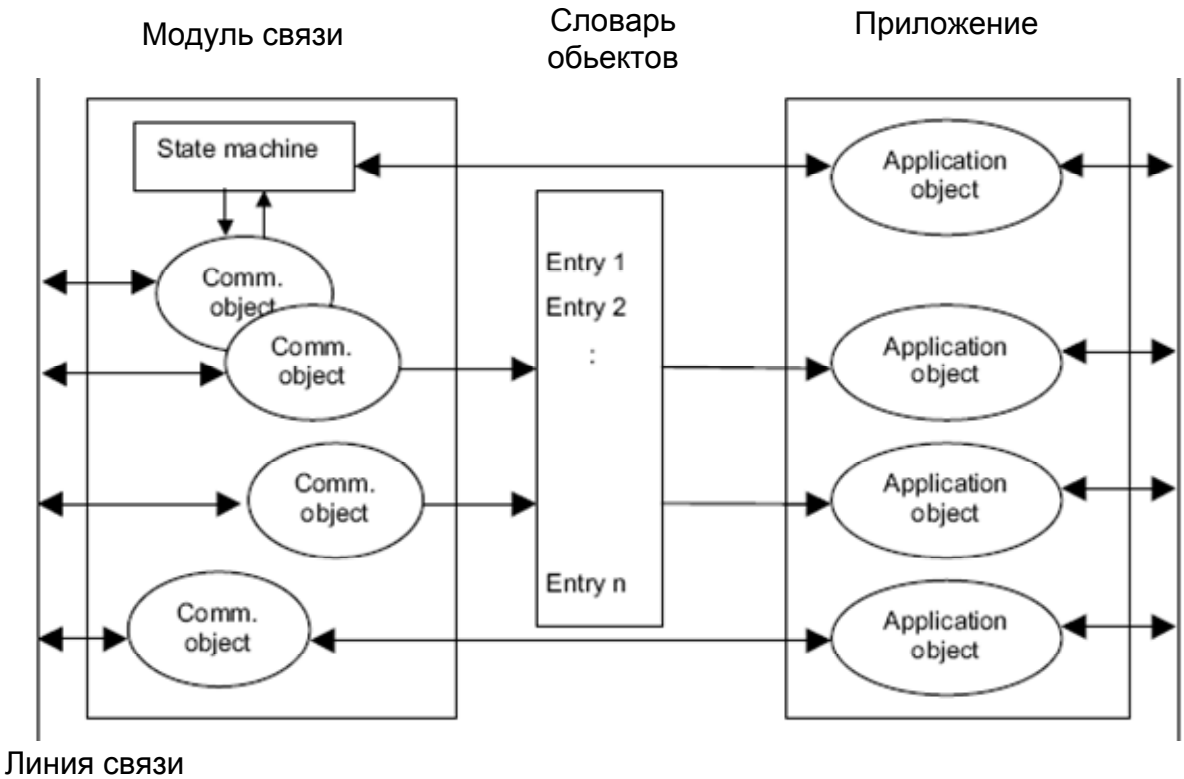


Рисунок 3. Обьектная модель.

- Модуль связи. Этот функциональный блок предоставляет коммуникационные обьекты и имеет соответствующую функциональность, для того чтобы транспортировать данные из этих обьектов через нижележащие слои информационной сети.

- Словарь обьектов. Отображает все элементы данных, которые могут оказывать влияние на выполнение приложений, все коммуникационные обьекты и автоматы состояний.
- Приложение – набор программ, которые обеспечивают функциональность устройства, в аспекте интерактивной работы с ним.

Таким образом, словарь обьектов является связующим звеном между коммуникациями и приложением. Полное описание приложения с точки зрения всех элементов данных указанных в словаре обьектов называется конфигурация устройства.

6.2.2 Словарь обьектов

Самая важная часть конфигурации – Словарь обьектов. Словарь обьектов – по существу, группа доступных через сеть, в предустановленном порядке обьектов. К каждому из обьектов словаря обращаются используя 16 разрядные индексы. Полное строение Словаря Обьектов показано на странице. Это построение близко соответствует другим концепциям построения промышленных последовательных шин.

Индекс(hex)	Обьект
0000	Не используется
0001-001f	Статические(простые) типы данных
0020-003f	Комплексные(составные) типы данных
0040-005f	Комплексные типы данных, определённые производителем
0060-007f	Статические типы данных определяемые профилем устройства.
0080-009f	Комплексные типы данных определяемые профилем устройства.
00a0-0fff	Зарезервировано
1000-1fff	Область параметров связи.
2000-5fff	Область профилей устройств, определяемых изготовителем.
6000-9fff	Область профилей стандартизованных устройств.
a000-bfff	Область стандартизованных интерфейсов профилей.
cfff-ffff	Зарезервировано

Обьектный словарь может содержать информацию о 65335 обьектах, адресуемых 16 битным индексом.

Статические типы данных с индексами 0001-001f содержат определения стандартных типов таких как, Булевы, Целые, с плавоющей точкой строка и тп. Эти ссылки введены исключительно в справочном порядке и не могут быть ни прочитаны, ни считаны.

Комплексные типы данных с индексами 0020-003f предопределённые структуры состоящие из стандартных типов данных, общие для всех устройств.

Комплексные типы данных, определённые производителем с индексами 0040-005f структуры состоящие из стандартных типов данных, но специфичные для конкретного устройства.

Конфигурация устройства может определять дополнительные типы данных специфичные для данного типа устройств. Статические типы данных, определённые профилем устройства внесены в список с индексами 0060-007f, комплексные с индексами 0080-009f

Устройство может опционально предоставлять допуск к чтению структур поддерживаемых комплексных типов данных, по соответствующему индексу. Под-индекс 0, при этом содержит количество элементов в этом индексе, а последующие под-индексы содержат закодированные числом UNSIGNED16 , типы в соответствии с таблицей 39.

Область параметров связи, с индексами 1000-1fff содержит специфические параметры необходимые для обмена по CAN-сетке.Эти элементы общие для всех устройств.

Область конфигураций стандартизованных устройств с индексами 6000-9fff содержит все обьекты данных устройства, общие для класса подобных устройств, которые могут быть записаны или прочитаны через сеть. Конфигурации устройств могут использовать элементы 6000-9fff, чтобы описать параметры устройства и его функциональные возможности. В границах этого диапазона могут быть описаны до 8 отдельных устройств. В этом случае такие устройства называются многофункциональными модулями. Такой модуль вмещает в себя до 8ми сегментов с конфигурациями различных устройств. Эта особенность позволяет создавать устройства с множественной функциональностью. Элементы конфигураций различных устройств сдвинуты относительно друг друга на 800h.

6000h to 67FFh device 0
6800h to 6FFFh device 1
7000h to 77FFh device 2
7800h to 7FFFh device 3
8000h to 87FFh device 4
8800h to 8FFFh device 5
9000h to 97FFh device 6
9800h to 9FFFh device 7

Распределение Обьектов Данных Процесса (PDO), для каждого сегмента многофункционального модуля должно быть осуществлено со смещением 64, то есть первый PDO второго сегмента начинается с адреса 65. В этой связи в системе поддерживается максимум 8 сегментов.

Концепция словаря обьектов допускает введение производителем дополнительных функциональных возможностей в свои устройства, но необходимо отразить это в словаре обьектов предопределённым образом. Пространство в словаре с индексами 2000-5fff определено для введённых изготовителем функциональных возможностей.

6.3 Модель взаимодействия.

Модель взаимодействия определяет различные коммуникационные обьекты и сервисы, и различные механизмы запуска и передачи сообщений. Модель взаимодействия поддерживает передачу синхронных и асинхронных сообщений. Посредством передачи синхронных сообщений, осуществляется координированный обмен данными, и возможно, управление. Синхроннизация передачи сообщений осуществляется предопределёнными коммуникационными обьектами(синхронизирующее сообщение, отметка времени). Синхронные сообщения передаются по факту появления предопределённого синхронизирующего сообщения. Асинхронные сообщения могут быть переданы в любое время.

Из-за характера событий основного механизма связи(CAN-bus), целесообразно определить время подавления повторной передачи одного и того же обьекта, для того чтобы гарантировать что не происходит подвисания передачи данных для низкоприоритетных обьектов. Время подавления для обьекта данных определяет минимальное время которое должно пройти между двумя последовательными вызовами сервисов передачи для данного обьекта. Интервал запрета назначается приложением. В зависимости от функциональных возможностей различают 3 модели отношений в сети.

- Мастер – слэйв
- Клиент – сервер
- Производитель потребитель

6.3.1 Мастер – слэйв

В любое время есть одно устройство – мастер и другие - его слэйвы, Мастер обращается к слэйвам. Те или просто рассматривают запрос как индикацию, либо отвечают, если того требует протокол.

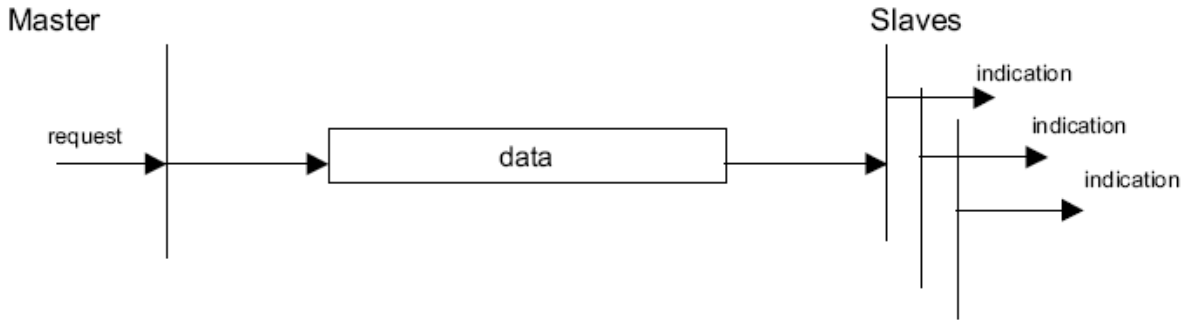


Рисунок 4. Мастер-слэйв коммуникация без подтверждения

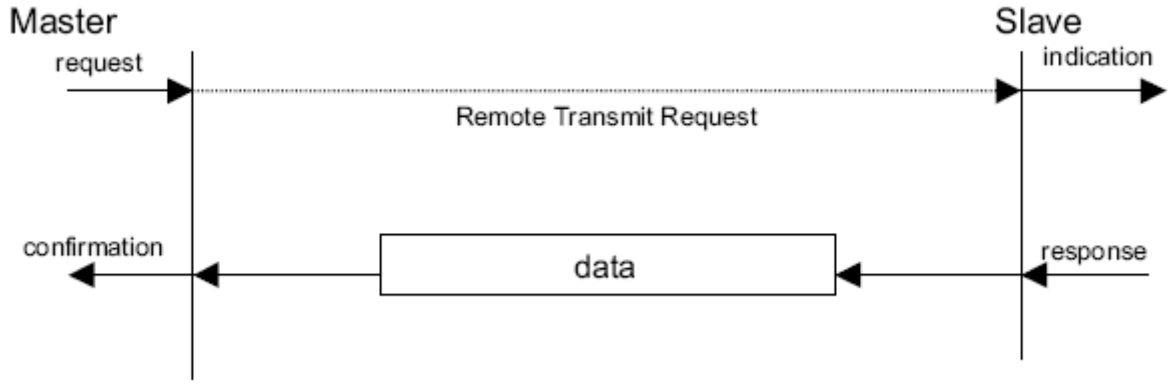


Рисунок 5. Мастер-слэйв коммуникация с подтверждением

6.3.2 Клиент – сервер

Это модель отношений между одним сервером и одним клиентом. От клиента исходит запрос на подкачку/загрузку данных, который запускает на сервере действия по выполнению заданной задачи, по завершении которой сервер отвечает на запрос.

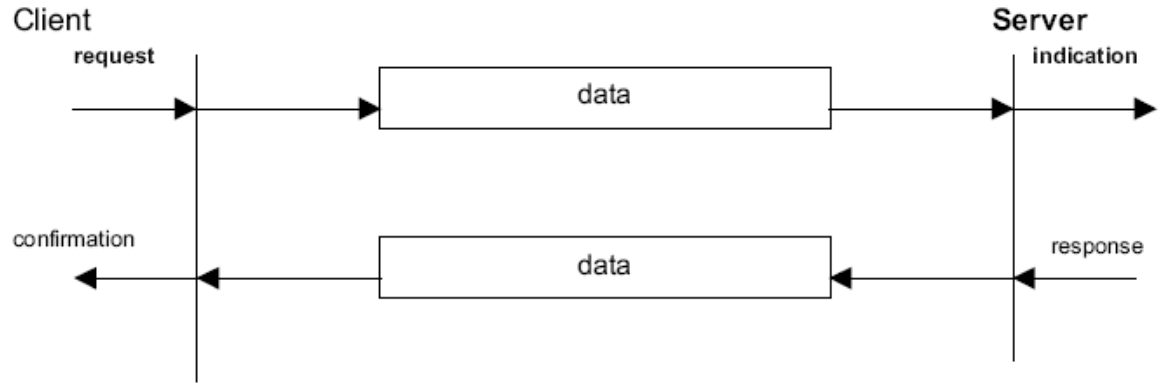


Рисунок 6. Клиент-сервер.

6.3.3 Производитель-потребитель.

Модель отношений производитель-потребитель предполагает наличие одного производителя и от нуля и более потребителей. Push модель характеризуется неподтверждаемым запросом исходящим от производителя. Pull модель характеризуется подтверждаемыми запросами потребителей к производителю.

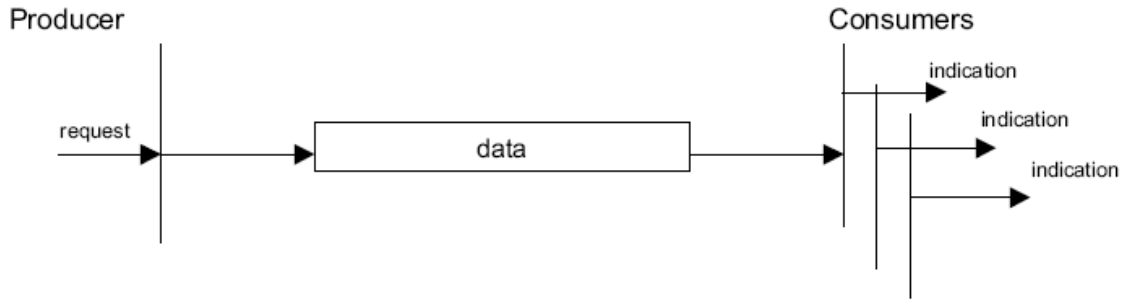


Рисунок 7. Push модель

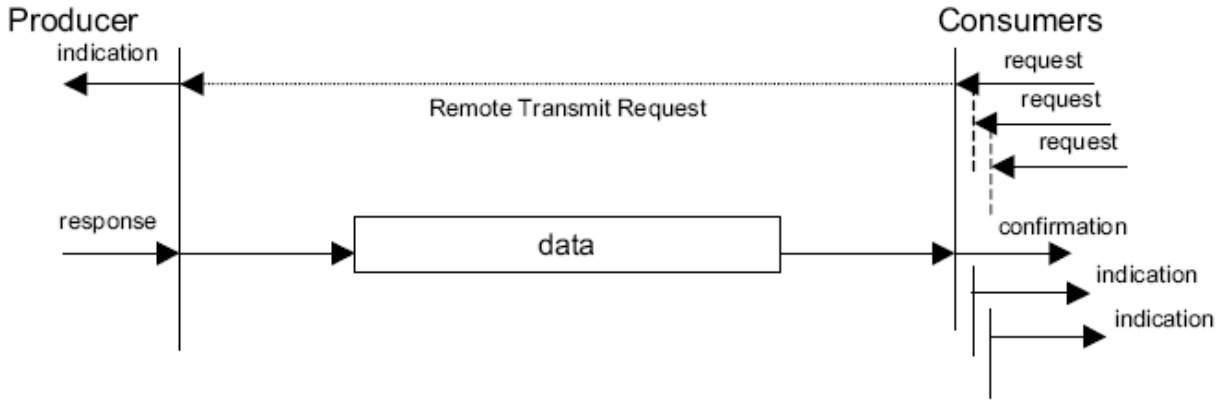


Рисунок 8. Pull модель.

9. Уровень приложений

9.1 Типы данных и правила кодирования.

9.1.1 Общее описание типов данных и правил кодирования.

Чтобы иметь возможность передачи многообразных данных через сеть CAN, формат этих данных и их смысл должен быть известен источнику этих данных и их получателю. Данная спецификация даёт образец этого, в соответствии с концепцией типов данных.

Правила кодирования определяют способы представления переменных различных типов данных, и синтаксис этого представления для CAN сети. Значения переменных представляются битовыми последовательностями. Битовые последовательности разбиваются для передачи в сети на последовательности байтов(октетов). Для кодировки целочисленных типов данных используется little endian стайл(ёу) , то есть идёт возрастание номеров передаваемых октетов начиная с самого младшего, как это показано на рисунке 9.

Часто приложениям, помимо базовых типов, требуются иные типы данных. Список имеющихся типов данных может быть расширен использованием механизма смешанных типов данных. В CAN open встроенный набор типов данных уже расширен за счёт таких широко распространённых типов данных как «Отображаемая строка» и «Дата и время»(см. пп. 9.1.6.2 и 9.1.6.4). Для определения пользователем составных типов данных используется в ключевое слово «DEFTYPES»(в терминологии данной спецификации) а не «DEFSTRUCTS»(см таблицу 37 «Словарь обьектов - Определение обьектов»)

9.1.2 Определение типа данных.

Тип данных задаётся как связь между значением данного типа и его кодированием. Имена типов данных задаются при определении этих типов. Синтаксис описания данных и определения типов данных такой как показано ниже(см EN EN 61131-3).

```
data_definition                ::=  type_name    data_name
определение_данных(обьекта)    есть имя_типа     имя_данных(обьекта)

type_definition                ::=   constructor type_name
определение_типа               есть  конструктор имя_типа

constructor                    ::=   compound_constructor |
                                     basic_constructor

конструктор                    есть  конструктор_составного_типа или
                                     конструктор_базового_типоа

compound_constructor           ::=   array_constructor |
                                     structure_constructor
конструктор_составного_типа    есть  конструктор_массивов или
                                     конструктор_структур

array_constructor              ::=   'ARRAY' '[' length ']''OF'type_name
конструктор_массивов           есть  'ARRAY' '[' длина ']''OF' имя_типа

structure_constructor          ::=   'STRUCT' 'OF' component_list
конструктор_структур           есть  'STRUCT' 'OF' список_компонентов

component_list                 ::=   component { ',' component }
список_компонентов             есть  компонент (',' компонент)

component                      ::=   type_name component_name
компонент                      есть  имя_типа имя_компонента
```

```
basic_constructor            ::=   'BOOLEAN' |
                                   'VOID' bit_size |
                                   'INTEGER' bit_size |
                                   'UNSIGNED' bit_size |
                                   'REAL32' |
                                   'REAL64' |
                                   'NIL'
конструктор_базового_типа    есть  'BOOLEAN' |
                                   'VOID' размер_в_битах |
                                   'INTEGER' размер_в_битах |
                                   'UNSIGNED' размер_в_битах |
                                   'REAL32' |
                                   'REAL64' |
                                   'NIL'

bit_size                     ::=   '1' | '2' | <...> | '64'
размер_в_битах               есть  от '1' до '64'

length                       ::=   positive_integer
длина                        есть  положительне_целое

data_name                    ::=   symbolic_name
имя_данных(обьекта)          есть  символьное_имя

type_name                    ::=   symbolic_name
имя_типа                     есть  символьное_имя


component_name               ::=   symbolic_name
имя_компонента               есть  символьное_имя

symbolic_name                ::=   letter {[ '_' ]( letter | digit )}
символьное_имя               есть  символ{['_'](символ или цифра)}
                                   произвольное количество символов или
                                   цифр, которое должно начинаться с
                                   символа

positive_integer             ::=   digit { digit }
положительное_целое          есть  цифра {цифра}
                                   произвольное количество цифр

letter                       ::=   'A' |
                                   'B' | <...> | 'Z' |
                                   'a' | 'b' | <...> | 'z'
символ                       есть  заглавные и строчные симвлоы
                                   латинского алфавита

digit                        ::=   '0' | '1' | <...> | '9'
цифра                        есть  цифра :)
```

Рекурсивные определения не допустимы, то есть нельзя задать тип – массив структур, или задать тип структуры, в состав которой входят другие структуры??.

Тип данных определённый в соответствии с `type_definition` называется базовым(и наоборот составным), если при определении этого типа используется один из базовых конструкторов(составных конструкторов).

9.1.3 Последовательности битов.

9.1.3.1 Определение битовой последовательности.

Бит, как известно, может принимать значение 0 или 1. Битовой последовательностью b назовём упорядоченный набор битов длиной от 0 до ∞. Если битовая последовательность включает в себя хотя бы один бит, то она обозначается bj, где j>0. Пусть у нас есть биты $b_0,b_1,...,b_{n-1}$, где число n – положительное целое. Тогда

$$b = b_0,b_1,...,b_{n-1}$$

назовём битовой последовательностью длиной |b|=n. Пустую битовую последовательность (т.е. длина которой – 0 бит), будем обозначать как e.

Примеры битовых последовательностей. 1001010,1001010, 1001010

Оператор(~) инверсии преобразует битовую последовательность

$$b_0,b_1,...,b_{n-1}$$

в битовую последовательность

$$\sim b_0,\sim b_1,...,\sim b_{n-1}$$

(математика гласит что `~0 == 1, a ~1==0`)

Основной операцией(помимо инверсии) над битовыми последовательностями является конкатенация или иными словами склейка. Пусть у нас имеется две битовых последовательности – $a = a_0,a_1,...,a_{m-1}$ и $b = b_0,b_1,...,b_{n-1,}$ тогда результатом конкатенации будет битовая последовательность

$ab = a_0,a_1,...,a_{m-1,}b_0,b_1,...,b_{n-1}$ длинной $|ab|=|a|+|b| = m+n$.

Пример конкатенации (11)(11) = (1111), замечу что для этой операции абсолютно неприемлем переместительный закон. И на последок – конкатенация любой битовой последовательности с пустой последовательностью даёт ту же самую последовательность

$$ae = ea = a$$

9.1.3.2 Синтаксис передачи данных для битовых последовательностей.

Для передачи через CAN сетку, последовательность битов переупорядочивается в последовательность октетов. Здесь и далее числовые значения октетов приводятся в шестнадцатеричной записи, то есть Eh. Пусть у нас имеется битовая последовательность $b_0,b_1,...,b_{n-1}$, где n<=64. Определим такое целое положительное число K, что 8*(K-1)<N<=8*K. Тогда получается что указанная битовая последовательность будет передана за K октетов упорядоченных как показано на рисунке 9. Биты старшего октета, номер которых превышает N-1 (то есть номер старшего бита в последовательности), не содержат битов из передаваемой последовательности, в общем случае они не определены. Как было сказано выше сначала передаются младшие байты, затем старшие. Байты передаются начиная со своих старших битов, отсюда наша последовательность будет передана в порядке:

$$b_7,b_6,...,b_{1,}b_0,b_{15},...,b_{9,}b_8,...$$

octet number	1.	2.	k.
	b_7 .. b_0	b_{15} .. b_8	b_{8k-1} .. b_{8k-8}

Рисунок 9 . Синтаксис передачи данных для битовых последовательностей

Пример Битовая последовательность b = b_0,...,b_9 = 0011 1000 01 представляющее число типа UNSIGNED10, и имеющее значение 021Ch будет передана в виде двух последовательных октетов первый 1Ch b второй 02h.

b_9, ... ,b_0
1 0 0 0 0 1 1 1 0 0
2h 1h Ch

9.1.4 Базовые типы данных.

Для базовых типов данных «type_name» совпадает с символьным именем соответствующего ему конструктора, например:

BOOLEAN BOOLEAN

есть определение типа для BOOLEAN типа данных.

9.1.4.1 NIL

Данные базового типа NIL представляют крокозябр ∈(Пустые множества битов).

9.1.4.2 Boolean

Данные базового типа BOOLEAN принимают значения TRUE или FALSE. Значения этой переменной представляются битовой последовательностью длиной 1. Значению TRUE соответствует битовая последовательность 1, значению FALSE – 0.

9.1.4.3. Void

Данные базового типа VOIDn представляются битовой последовательностью длиной n бит. Значения данных для типа VOIDn не определяются. Ьиты в последовательности, для типа VOIDn должны быть либо определены явно, либо помечены как «do not care». Данные типа VOIDn могут использоваться либо для зарезервированных полей, либо для выравнивания переменных в смешанных типах данных на границу байтов.

9.1.4.4 Целочисленные без знаковые.

Данные базового типа UNSIGNEDn имеют значения неотрицательных целых чисел. Диапазон принимаемых значений 0, …, 2^n-1. Данные представляются битовой последовательностью длиной n. Битовой последовательности

$$b = b_0,b_1,...,b_{n-1}$$

соответствует значение UNSIGNEDn(b) = $b_{n-1} * 2^{n-1} + \ldots + b_1 * 2^1 + b_0 * 2^0$.
Заметим ещё раз, что битовая последовательность начинается с самого младшего октета. Пример. значение 256 = 100h данных типа UNSIGNED16 будет передана через шину в виде двух октетов – первый 00 h , второй 01h.

Следующие типы данных UNSIGNEDn будут переданы как это показано в таблице.

octet number	1.	2.	3.	4.	5.	6.	7.	8.
UNSIGNED8	$b_7..b_0$							
UNSIGNED16	$b_7..b_0$	$b_{15}..b_8$						
UNSIGNED24	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$					
UNSIGNED32	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$				
UNSIGNED40	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$			
UNSIGNED48	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$		
UNSIGNED56	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	
UNSIGNED64	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	$b_{63}..b_{56}$

Рисунок 10.Синтаксис передачи для типа данных UNSIGNEDn

9.1.4.5 Знаковые целочисленные.

Данные базового типа INEGERn имеют значения целых чисел. Диапазон принимаемых значений -2^{n-1}, …, $2^{n-1}-1$. Данные представляются битовой последовательностью длиной n. Битовой последовательности

$$b = b_0,b_1,...,b_{n-1}$$

соответствует значение

$$INTEGERn(b) = b_{n-2} * 2^{n-2} + \ldots + b_1 * 2^1 + b_0 * 2^0, \text{ если } b_{n-1}=0.$$

и произведя арифметическое дополнение до двух

$$INTEGERn(b) = - INTEGERn(\sim b) - 1, \text{ если } b_{n-1}=1.$$

Заметим ещё раз, что битовая последовательность начинается слева с самого младшего бита. Пример. значение -256 = FF00h данных типа INTEGER16 будет передана через шину в виде двух октетов – первый 00 h , второй FFh.

Следующие типы данных INTEGERn будут переданы как это показано в таблице.

octet number	1.	2.	3.	4.	5.	6.	7.	8.
INTEGER8	$b_7..b_0$							
INTEGER16	$b_7..b_0$	$b_{15}..b_8$						
INTEGER24	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$					
INTEGER32	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$				
INTEGER40	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$			
INTEGER48	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$		
INTEGER56	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	
INTEGER64	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	$b_{63}..b_{56}$

Рисунок 11.Синтаксис передачи для типа данных INTEGERn

9.1.4.6 Числа с плавающей точкой.

Данные базового типа REAL32 и REAL64 принимают значения вещественных чисел.

Данные типа REAL32 представляются битовой последовательностью длиной 32 бита. Кодирование значений соответствует стандарту IEEE 754-1985 для чисел с плавающей точкой одинарной точности.

Данные типа REAL64 представляются битовой последовательностью длиной 64 бита. Кодирование значений соответствует стандарту IEEE 754-1985 для чисел с плавающей точкой двойной точности.

Последовательность битов длиной 32 бита может либо иметь значение (конечное, отличное от нуля вещественное значение, ±0, ±∞) либо быть NaN(not a number). Битовой последовательности

$$b = b_0,b_1,...,b_{31}$$

соответствует значение(в случае конечного ненулевого вещественного числа)

$$REAL32(b) = (-1)^{S} * 2^{E-127} * (1 + F),$$

где
$S = b_{31}$, знак
$E = b_{30} * 2^{7} + \ldots + b_{23} * 2^{0}$, $0 < E$ **< 255**, не смещённая экспонента(т.е. беззнаковая)
$F = 2^{23} * (b_{22} * 2^{22} + \ldots + b_{1} * 2^{1} + b_{0} * 2^{0})$ мантисса числа.

Заметим ещё раз, что битовая последовательность начинается слева с самого младшего бита.

Пример.

$6.25 = 2^{E-127}(1 + F)$ где
$E = 129 = 2^{7} + 2^{0}$, а
$F = 2^{-1} + 2^{-4} = 2^{-23}(2^{22}+2^{19})$ таким образом, число будет представлено как последовательность битов:

S	*E*	*F*
b_{31}	*b_{30} .. b_{23}*	*b_{22} .. b_0*
0	*100 0000 1*	*100 1000 0000 0000 0000 0000*

6.25 = b0,...,b31 = 0000 0000 0000 0000 0001 0011 0000 0010

Эта последовательность будет передана в следующем порядке.

octet number	**1.**	**2.**	**3.**	**4.**
REAL32	00h	00h	C8h	40h
	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$

9.1.5 Составные типы данных.

Определение составных типов данных восходит к уникальному списку определений типов, включающему в себя только базовые типы данных. Соответственно данные составного типа с именем ‘type_name’ есть упорядоченные списки данных-копонент с именами ‘имя_компоненты_i’, базовых типов ‘базовый_тип_i’

Конструкторы смешанных типов данных – ARRAY и STRUCT OF.

```
STRUCT OF

      базовый_тип_1 имя_компоненты_1,
      базовый_тип_2 имя_компоненты_2,
      … …
      базовый_тип _N имя_компоненты_N

имя_составного_типа

ARRAY [длина] OF базовый_тип имя_составного_типа
```

Битовая последовательность представляющая составной тип данных получается путём конкатенации битовых последовательностей представляющих собой данные компонент.

Принимая, что компоненты с именами ‘имя_компоненты_i’ представляются битовыми последовательностями.

$$b(i)\ ,\ \text{где}\ i = 1, \ldots, N$$

Тогда составной тип данных будет представлен конкатенированной последовательностью

$$b_0(1),\ldots,b_{n1-1}(1),\ldots,b_{nN-1}(N)$$

Пример.
Рассмотрим тип данных

```
STRUCT OF
      INTEGER10      x,
      UNSIGNED5      u
NewData
```

Примем x = -423 = 259h и u = 30 = 1Eh. Пусть b(x) и b(u), означают битовые последовательности представляющие битовые последовательности величин x и u, соответственно. Тогда:

```
b(x) = b0(x) .. b9(x) = 1001101001
b(u) = b0(u) .. b4(u) = 01111
b(xu) = b(x) b(u) = b0(xu) .. b14(xu) = 1001101001 01111
```

b_0 b_7 b_8 b_{14}

1001 1010 0101 111

9h 5h Ah 7h(так как октеты передаются начиная со старших битов, то 1010 следует читать задом наперёд)

Значение структуры будет передано двумя октетами $b_7 \ldots b_0$ первый (59h), b14 … b8 второй (7Ah).

9.1.6 Расширенные типы данных.

Расширенные типы данных содержат в себе базовые типы данных, и описаны в следующих подпунктах.

9.1.6.1 Строка байт(или строка октетов)

В определении типа OCTET_STRINGlenght length означает длину строки в октетах(байтах).

```
ARRAY [ length ] OF UNSIGNED8 OCTET_STRINGlength
```

9.1.6.2 Видимая строка.

Допустимые значения октетов в этой строке 00h и в диапазоне от 20h до 7Eh. Символы строки закодированы в соответствии с ISO 646-1973(E) как 7ми битовые символы. Длина строки задаётся в октетах.

```
UNSIGNED8 VISIBLE_CHAR
ARRAY [ length ] OF VISIBLE_CHAR VISIBLE_STRINGlength
```

0h требуется для указания конца строки.

9.1.6.3 Строка Юникод

Длина в октетах.

```
ARRAY [ length ] OF UNSIGNED16 UNICODE_STRINGlength
```

9.1.6.4 Дата и время.

Данное типа Дата и время указывается в абсолютном значении миллисекунд и дней. Миллисекунды содержат количество миллисекунд прошедших с полуночи. Дни показывают количество дней прошедших с 1 января 1984 года. Длительность битовой последовательности, которой кодируется данная структура – 48 бит

```
STRUCT OF
      UNSIGNED28 ms,
      VOID4 reserved,
      UNSIGNED16 days
TIME_OF_DAY
```

9.1.6.5 Временная разница.

Тип разницы во времени содержит такие же поля, что и Дата и время, разница хранится в виде разностей в миллисекундах и в днях. Длительность битовой последовательности, которой кодируется данная структура – 48 бит

```
STRUCT OF
      UNSIGNED28 ms,
      VOID4 reserved,
      UNSIGNED16 days
TIME_OF_DAY
```

9.1.6.6 Блок данных

Блок данных служит для передачи произвольного большого блока данных от клиента к серверу и наоборот. Структура внутреннего содержимого таких блоков информации определяется приложением и не рассматривается в этом документе.

9.2. Обьекты связи (Коммуникационные обьекты)

Обьекты связи COBs описываются сервисами и протоколами. Все сервисы описываются табличным способом, где указываются параметры каждого сервисного примитива определённого для данного сервиса. Примитивы, определённые для данного отдельного сервиса, задают тип сервиса (т.е. подтверждаемый, неподтверждаемый и тд.) Как интерпретируется эта табличная форма и какие типы сервисов существуют, уже рассказывалось в п 6.3.

При работе сервисов предполагается, что не происходит никаких сбоев в физическом уровне CAN сети. Такие вопросы должны решаться в рамках приложений.

9.2.1 Обьекты данных процесса.(PDO)

Передача данных в реальном времени осуществляется посредством обьектов данных процесса PDO. Передача PDOsов производится широковещательно, без дополнительных издержек протокола.

PDOsы описываются элементами в Обьектном Словаре устройства, и предоставляют доступ к обьектам приложений. Типы данных и отображение обьектов приложений в PDO, определяется структурой предопределённого типа (PDO-отображение, PDO-mapping) в обьектном словаре устройства. Если поддерживается изменение PDO-отображения, то количество PDOsов и отображенеие обьектов приложений внутри них могут быть загружены в устройство на стадии процесса конфигурации устройства (см раздел Процедура инициализации), с помощью использования SDO сервисов для соответствующих элементов Обьектного словаря.

Количество и длина PDOsов устройства индивидуальны для разных приложений, и должны быть определены в профиле устройства.

Имеется 2 вида использования PDOsов – передача данных(TPDOs) и приём данных(RPDOs). Устройства поддерживающие TPDOs называются PDO производителями, а устройства обладающие возможностью принимать PDO называются PDO потребителями. PDOsы описываются параметром связи PDO(20h) и параметром отображения PDO(21h). Структура этих данных объяснена в п 9.5.4. Параметр связи описывает коммуникационные возможности PDO, структурный параметр определяет содержимое PDOsa(переменные устройства). Индексы соответствующих элементов Словаря обьектов вычисляются по приведённым правилам.

- Индекс RPDO параметра связи = 1400h + $RPDO_{Number}$ - 1.
- Индекс TPDO параметра связи = 1800h + $TPDO_{Number}$ - 1.
- Индекс RPDO структурного параметра = 1600h + $RPDO_{Number}$ - 1.
- Индекс TPDO структурного параметра = 1A00h + $TPDO_{Number}$ - 1.

Пара описанных параметров обязательна для каждого PDO. Упомянутые элементы описаны в главе 9.5.

9.2.1.1 Режимы передачи.

Бывают:

- асинхронная
- синхронная

В целях синхронизации в сеть приложением синхронизации периодически передаются специальные синхрособщения(SYNC). SYNC обьекты являются предопределёнными коммуникационными обьектами (COBjами) см пункт 9.3.3. На рисунке 13 показан принцип синхронной и асинхронной передач. Синхронные PDOsы передаются внутри предопределённого временного интервала непосредственно после SYNC обьекта. Синхронная передача данных описывается в главе 9.3.

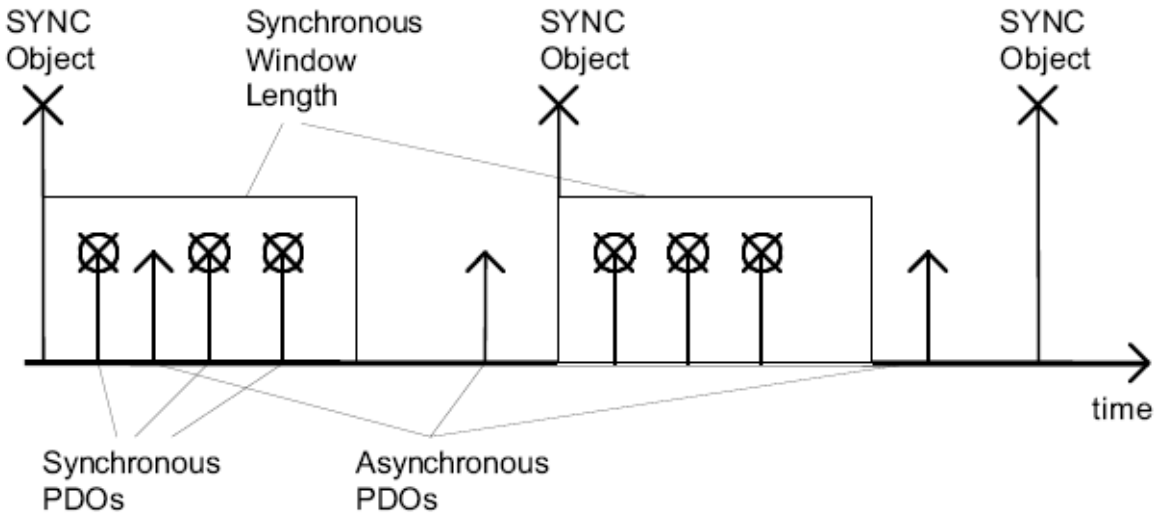


Рисунок 13 . Синхронная и асинхронная передача данных. Как видно из рисунка асинхронные PDOsы могут прерывать даже передачу последовательности синхронных(но уже начатый синхронный будет передан).

Параметр типа передачи PDOбъекта определяет режим передачи и способ запуска передачи.

Для синхронных TPDosов способ передачи определяется так же коэффициент пропорциональности запуска относительно синхросообщений. Если задаётся коэффициент 0, то передача будет ациклично(т.е. один раз) выполнена после ближайшего прохождения синхросообщения. Если коэффициент 1, то передача будет осуществляться после каждого синхросообщения. Если коэффициент N, то передача будет осуществляться каждый Nный раз. Асинхронные TPDosы передаются безо всякой связи с синхрообьектами. Данные полученные синхронным RPDosом попадают приложению сразу после прихода следующего синхросообщения. независимо от коэффициента кратности запусков. Данные асинхронного RPDosa попадают непосредственно приложению (точас же. ?(видимо должен быть метод реакции на такое прерывание).).

9.2.1.2 Режимы вызова

Различают 3 режима вызова сообщений.

- **По событию.**

Механизм передачи запускается по возникновении внутренних событий в приложении , к которому относится COBj , Для синхронного это истечение заданного интервала времени. Для ациклично передаваемого синхронного и асинхронных сообщений, вызов передачи порождается какими либо событиями в приложении, и описывается в описании устройства.

- **То таймеру**

Передача сообщения происходит либо по внутреннему событию либо если такое событие не наступило.

- **Удалённый запрос.**

Передача асинхронного PDO запускается при получении удалённого запроса от какого либо устройства(PDO потребителем)

.

9.2.1.3 PDO Службы.

Как передача данных подчинена модели отношений производитель – потребитель было рассказано в п 6.3.3. Атрибуты PDOsa

- Номер — Номер обьекта PDO [1..512] для каждого пользовательского типа в локальном устройстве.

-Пользовательский тип	Одно из значений {consumer, producer}(потребитель-производитель)
-Время подавления обьекта.	n*100 us, n>>0.

9.2.1.3.1 Запись PDO

Для записи PDO используется Push модель (см п 6.3.3). На произвольное количество потребителей приходится один производитель данных.

Через эту службу производитель посылает свои данные указаным в его??? описании потребителям.

Таблица 3 Запись PDO

параметр	запрос/индикация
Аргумент	**Наличие**
Номер PDO	Обязательно
Данные	Обязательно

9.2.1.3.2 Чтение PDO

Чтение PDO производится по Pull модели (см п 6.3.3). Через этот сервис потребители нуждающиеся в PDO запрашивают получение указаной у производителя информации. Этот сервис с подтверждением Подтверждением запроса здесь является полученный удалённый обьект.

Таблица 4 Чтение PDO

параметр	запрос/индикация	ответ/подтверждение.
Аргумент	**Наличие**	**Наличие**
Номер PDO	Обязательно	
Удалённый результат		
Данные		Обязательно

9.2.1.4. Протоколы передачи PDO

9.2.1.4.1 Протокол записи PDO

Запись PDO это неподтверждаемый запрос. Производитель PDO посылает их в сетку. Может быть произвольное количество потребителей(от 0 и более). Приём потребителем данных называется индикация.

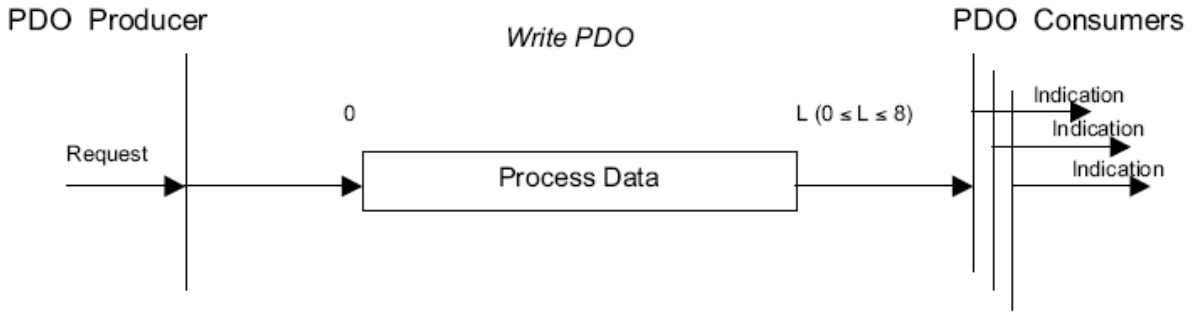
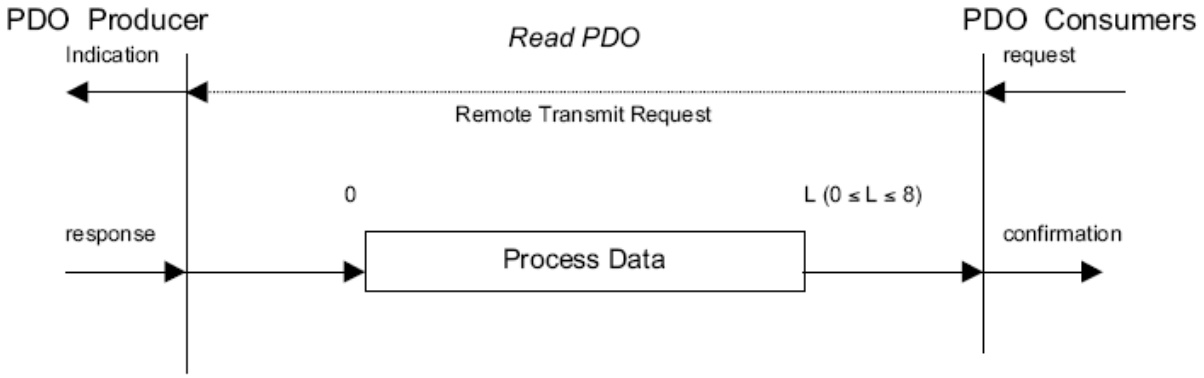


Рисунок 14 . Протокол записи PDO.

Описание: До L байт данных вмещает PDO согласно своему описанию. Если количество байт L оказывается больше чем N байт, которые указаны в описании, то реально используются только N байтов указанных в описании. Если информации в PDO оказывается меньше, то принятые данные не обрабатываются и передаётся Аварийный обьект(EMCY) с кодом 8210h если установлен обработчик аварийных ситуаций. ???.

9.2.1.4.2 Протокол чтения PDO

Запись PDO это запрос с подтверждением. Один или несколько(??? Видимо неодновременно) PDO посылают в сеть удалённые запросы обьекта.(transmission request frame - RTR). По приёме RTR производитель PDO, передаёт запрошенный обьект.



Описание: До L байт данных вмещает PDO согласно своему описанию. Если количество байт L оказывается больше чем N байт, которые указаны в описании, то реально используются только N байтов указанных в описании. Если информации в принятом потребителем PDO оказывается меньше, то принятые данные не обрабатываются и выдаётся аварийный код 8210h если установлен обработчик аварийных ситуаций. ???.Где?

9.2.2. Сервисные обьекты данных(SDO).

С помощью сервисных обьектов данных обеспечивается доступ к элементам Словаря Обьектов(изменение элементов). Так как эти элементы могут содержать данные произвольного размера и типы данных сервисных обьектов, SDOsы могут использоваться для передачи от клиента к серверу и обратно, множественных наборов данных (каждый из которых может содержать блок данных произвольного размера). Клиент может с помощью мультиплексора (индекс и подиндекс обьектного словаря) указывать какой из наборов данных будет передаваться. Содержимое наборов данных определено в этом же словаре.

В основном SDO передаётся как последовательность сегментов длинной до 127 сегментов – это есть сегмент-трансферт(segment transfer). Перед транспортировкой

сегментов есть стадия подготовки, когда клиент и сервер готовятся к обмену. Если требуется передать обьект размером до 4х байт, то обмен может быть произведён ещё на стадии инициализации. Этот механизм называется ускоренный трансферт(expedited transfer).
Дополнительно, SDO может передаваться как набор блоков, каждый размером до 127 сегментов, каждый из которых вмещает данные и порядковый номер сегмента в блоке. В стадии подготовки к передаче блока клиент и сервер оговаривают количество сегментов на которые разбит блок. После передачи блока есть завершающая фаза, во время которой клиент и сервер могут дополнительно проверить корректность произведённого обмена, путём проверки контрольных сумм . Описанный тип называется блок-трансферт(block transfer), и является для больших объёмов информации более быстрым типом обмена, по сравнению с чисто сегментным способом передачи(так как квитируется не каждый сегмент, а блок).

При выгрузке SDO блока может оказаться, что размер набора данных мал и не оправдывает его передачу блок-трансфертом из за издержек протокола. В таком случае на стадии инициализации перекачки может быть подключена поддержка автоматического перехода к сегментному или даже ускоренному трансферту, в зависимости от размера данных. Поскольку размер набора данных, при котором блочная передача предпочтительней сегментной или ускоренной, зависит от многих факторов, то клиент на стадии инициализации указывает серверу границу предпочтимости в байтах.

При блок-трансферте для подтверждения корректности передачи каждого блока, используется схема Go-Back-N ARQ (Автоматический запрос повторной передачи с откатом на N).

После загрузки блока на сервер, он сообщает клиенту последний успешно принятый сегмент в этом блок-трансферте, передавая ему порядковый номер этого сегмента. Делая это, сервер по умолчанию квитирует все сегменты предшествующие этому сегменту. Клиент начнёт следующий блок-трансферт с повторной передачи всех непроквитированных сегментов. Дополнительно к этому ***сервер*** должен указать количество сегментов в блоке для следующего блок-трансферта.

После выгрузки блока с сервера, клиент указывает последний успешно принятый сегмент в этом блок-трансферте, передавая ему порядковый номер этого сегмента. Делая это, клиент по умолчанию квитирует все сегменты предшествующие этому сегменту. Дополнительно к этому ***клиент*** должен указать количество сегментов в блоке для следующего блок-трансферта. ***(Тот кто получает информацию, тот и указывает сколько нужно получить сегментов, логично так как имеются физические ограничения на размер буфера приёма.)***

Для всех типов трансфертов клиент является инициатором начала обмена. Владелец Обьектного словаря, к которому осуществляется доступ, называется сервер SDO. И клиент и сервер могут быть инициаторами аварийного завершения передачи.

Посредством SDO, между двумя устройствами устанавливается одноранговый (точка-точка) канал связи. Устройство может поддерживать более чем один SDO. Один SDO-сервер(SSDO) поддерживается всегда, он называется SDO по умолчанию (Default SDO).

SDO описывается параметром связи SDO(тип описан в элементе с индексом 22h). Структура этого типа данных расписана в пункте 9.5.4. В двух словах, параметр связи SDO описывает коммуникационные настройки SDO, для SDO-сервера(SSDO) и SDO-клиента (CSDO) индексы доступа вычисляются по формулам:

- Индекс SSDO параметра связи = 1200h + НомерSSDO – 1
- Индекс CSDO параметра связи = 1280h + НомерCSDO – 1

Для каждого SDO наличие параметра связи обязательно. Только в случае если устройстве создан единственный SSDO, параметры связи могут быть пропущены. Означенные элементы расписаны в главе 9.5(Обьектный словарь).

9.2.2.1 SDO сервисы

Модель взаимодействия для SDO – есть модель клиент-сервер, описанная в пункте 6.3.2. SDO сервисы есть функции, используя которые клиент обращается(удалённо) к данным SDO-сервера.

Атрибуты:

- Номер SDO;	Число [1..128] определяет номер SDO в локальном устройстве.
- Тип пользователя;	Одно из значений {client, server}
- Мультиплексор типа данных	Мультиплексор содержит индекс и подиндекс и имеет тип STRUCTURE OF UNSIGNED (16) , UNSIGNED (8). Индекс определяет элемент словаря обьектов, а подиндекс указывает компонент, означенного элемента словаря.
- Тип трансферта;	Определяется длиной данных для трансферта: укороченный(expedited) для данных длиной менее 4х байт и сегментный или блочный(segmented or block) для данных длиной более 4х байт.
- Тип данных	Определяется мультиплексором.

В зависимости от прикладных требований, к SDO могут быть применены нижеприведённые сервисы:

- Загрузка SDO на сервер(download), которая разделяется на:

  - Инициация загрузки SDO.

  - Загрузка сегмента SDO.

- Выгрузка SDO с сервера(upload), которая разделяется на:

  - Инициация выгрузки SDO.

  - Выгрузка сегмента SDO.

- Аварийное завершение трансферта SDO.

При использовании сегментированной передачи SDO ПО связи ответственно за передачу SDO как последовательности.

В любом случае ПО должен поддерживаться ускоренный(expetited) трансферт. Сегментный трансферт должен поддерживаться если поддерживаются данные размером более 4х байт.

Опционно, могут реализовываться следующие SDO сервисы, реализующие блок-трансферт, который повышает пропускную способность шины, в случае необходимости передавать большие наборы данных:

- Блочная загрузка SDO на сервер, которая разбивается на :

  - Инициация загрузки блока.

  - Загрузка блока.

  - Окончание загрузки блока.

- Блочная выгрузка SDO блока из сервера, которая разбивается на:

  - Инициация выгрузки блока.

  - Выгрузка блока

  - Окончание выгрузки блока.

При использовании сервисов загрузки и выгрузки SDO, ПО связи отвечает за передачу данных как последовательности блоков. В протоколе «Блочная выгрузка(upload) SDO» должна осуществляться поддержка к переключению на протокол «Выгрузка SDO» (по сегментному или укороченному сценарию) на этапе Инициация выгрузки блока, для данных, размер которых не оправдывает использование блочного варианта из за издержек блочного протокола. Для аварийного завершения блочной передачи используется сервис Аварийное завершение трансферта SDO(см выше, не блочная передача).

9.2.2.1.1. Загрузка SDO(download).

При помощи этого сервиса клиент SDO осуществляет загрузку данных на сервер(сервером является владелец Обьектного словаря). Для сервера, при этом передаются собственно данные, мультиплексор(см. выше) набора данных, который будет загружаться, и его размер(для не блочного варианта этот параметр необязателен). Этот сервис квитируемый. Параметр Удалённый результат показывает успешно или нет обработан запрос. При неудаче, сервер опционно может возвратить причину отказа в загрузке(для отработки отказа на каждом этапе вместо подтверждения высылается запрос Аварийное завершение SDO трансферта??).

Процесс загрузки SDO включает в себя, как минимум, сервис Инициация загрузки SDO, для размера данных более 4х байт добавляется сервис Загрузка SDO сегмента.

Параметр	Запрос/Индикация	Ответ/Подтверждение
Аргумент	**Обязателен**	
Номер SDO	Обязателен	
Данные(порядок следв??)	Обязателен	
Размер	Необязателен	
Мультиплексор	Обязателен	
Удалённый результат		**Обязателен**
Успех		Как вариант
Отказ		Как вариант
Причина		Необязателен

Таблица 5. Загрузка SDO.

9.2.2.1.2 Инициация SDO загрузки.

При помощи этого сервиса клиент SDO указывает серверу, чтобы он приготовился к загрузке данных на сервер. Произвольно клиент может указать размер данных предназначенных к загрузке. Так же серверу указываются мультиплексор набора данных в который будет происходить загрузка и тип трансферта. В случае укороченного варианта загрузки, серверу указываются данные для набора данных, описываемого мультиплексором, сам мультиплексор и размер.

Параметр	Запрос/Индикация	Ответ/Подтверждение
Аргумент	**Обязателен**	
Номер SDO	Обязателен	
Размер	Необязателен	
Мультиплексор	Обязателен	
Тип трансферта	Обязателен	
Нормальный	Как вариант	
Укороченный	Как вариант	
Данные	Обязателен	
Удалённый результат		**Обязателен**
Успех		Обязателен

Таблица 6. Инициация SDO загрузки.

Данный сервис квитируемый. Параметр Удалённый результат указывает на успешную инициацию. В случае отказа обслуживания вместо подтверждения послается запрос Аварийное завершение трансферта SDO. В случае успешной укороченной загрузки помеченной как DOMAIN этот сервис производит загрузку данных, как это указано мультиплексором??.

9.2.2.1.3. Загрузка SDO сегмента.

При помощи этого сервиса клиент посылает данные в следующий сегмент сервера. Для сервера указываются данные и, опционно, их размер. Параметр продолжения показывает серверу, будут ли далее ещё загружаться сегменты, или этот сегмент последний в загрузке.

Параметр	Запрос/Индикация	Ответ/Подтверждение
Аргумент	**Обязателен**	
Номер SDO	Обязателен	
Данные	Обязателен	
Размер	Необязателен	
Параметр продолжения	Обязателен	
Загрузка не завершена	Как вариант	
Загрузка завершена	Как вариант	
Удалённый результат		**Обязателен**
Успех		Обязателен

Таблица 7. Загрузка SDO сегмента.

Данный сервис квитируемый. Параметр Удалённый результат указывает на успешную инициацию. В случае отказа обслуживания сервер должен послать запрос Аварийное завершение трансферта SDO. В случае успеха сервер добавляет полученный сегмент данных к загружаемому набору данных, и приготовится к добавлению следующего сегмента.

Одновременно на выполнении может быть одна загрузка SDO сегмента(в сети?? ИЛИ ДЛЯ СЕРВЕРА??).

Выполнению этого сервиса должен предшествовать успешный сервис Инициация SDO загрузки.

9.2.2.1.4. Выгрузка SDO.

При помощи этого сервиса клиент выгружает (upload) данные с сервера. Серверу указывается мультиплексор. Процесс выгрузки SDO включает в себя, как минимум, сервис Инициация выгрузки SDO, для размера данных более 4х байт добавляется сервис Выгрузка SDO сегмента(данные длиннее 4х байт).

Параметр	Запрос/Индикация	Ответ/Подтверждение
Аргумент Номер SDO Мультиплексор	**Обязателен** Обязателен Обязателен	
Удалённый результат Успех Данные Размер Отказ Причина		**Обязателен** Как вариант Обязательны. Необязателен Как вариант Необязателен

Таблица 8. Выгрузка SDO.

Этот сервис подтверждаемый. Возвращаемые данные указанны в таблице. Для отработки отказа на каждом этапе вместо подтверждения высылается запрос Аварийное завершение SDO трансферта.

9.2.2.1.5. Инициация SDO выгрузки.

Клиент запрашивает сервер выгрузить ему данные и указывает SDO номер и мультиплексор.

Параметр Удалённый результат указывает на успешную инициацию. В случае отказа сервер пришлёт клиенту вызов Аварийное завершение SDO передачи??. В случае успешной укороченной выгрузки сервер включит в ответ запрошенные данные.

Параметр	Запрос/Индикация	Ответ/Подтверждение
Аргумент Номер SDO Мультиплексор	**Обязателен** Обязателен Обязателен	
Удалённый результат Успех Размер Мультиплексор Тип трансферта Нормальный Укороченнный Данные		**Обязателен** Обязателен Необязателен. Обязателен Обязателен Как вариант Как вариант Обязателен.

Таблица 9. Инициация SDO выгрузки.

9.2.2.1.6 Выгрузка SDO сегмента.

Клиент каждый раз посылает запрос серверу, отвечая(удалённый результат – всегда успех) на который тот передаёт помимо сегмента данных, параметр продолжения (есть ли ещё сегменты) и размер данных если сегмент заполнен не полностью ??. . В случае отказа сервер пришлёт клиенту вызов Аварийное завершение SDO передачи??. Выполнению этого сервиса должен предшествовать успешный сервис Инициация SDO выгрузки.

Параметр	Запрос/Индикация	Ответ/Подтверждение
Аргумент	**Обязателен**	
Номер SDO	Обязателен	
Мультиплексор	Обязателен	
Удалённый результат		**Обязателен**
Успех		Обязателен
Размер		Необязателен.
Данные		Обязателен
Параметр продолжения		Обязателен
Загрузка не завершена		Как вариант
Загрузка завершена		Как вариант.

Таблица 10. Выгрузка SDO сегмента.

9.2.2.1.7. Аварийное завершение SDO передачи

Этот сервис прерывает SDO загрузки и выгрузки, указывая на них номером?? мультиплексором??. Опционно может быть указана причина отказа. Эт от сервис неподтверждаемый. Этот сервис может быть вызван в любое время как клиентом так и сервером. Если клиент ожидает квитирования сервиса, то индикация об Аварийном завершении SDO передачи рассматривается как квитирование.

Параметр	Запрос/Индикация
Аргумент	**Обязателен**
Номер SDO ???	Обязателен
Мультиплексор	Обязателен
Причина	Обязательн

Таблица 11. Аварийное завершение SDO передачи.

9.2.2.1.8. Блочная загрузка SDO на сервер.

При помощи данного сервиса клиент загружает данные на сервер используя протокол блочной загрузки. Серверу передаются данные, мультиплексор набора данных, куда будет идти загрузка и опционно размер. Это подтверждаемый сервис. Параметр удалённого результата может принимать значение успех или отказ. В случае отказа может быть указана причина.

Параметр	Запрос/Индикация	Ответ/Подтверждение
Аргумент Номер SDO Мультиплексор Данные Размер	**Обязателен** Обязателен Обязателен Обязателен Необязателен	
Удалённый результат Успех Отказ Причина		**Обязателен** Как вариант Как вариант Необязателен.

Таблица 12. Блочная SDO загрузка.

9.2.2.1.9. Инициация блочной SDO загрузки.

При помощи этого сервиса клиент SDO указывает серверу(владельцу Обьектного словаря), чтобы он приготовился к загрузке данных на сервер. Произвольно клиент может указать размер данных предназначенных к загрузке. Так же серверу указываются мультиплексор набора данных в который будет происходить загрузка и указание о проверке контрольных сумм по окончанию передачи блока. Клиент и сервер показывают друг другу способность и готовность своего ПО произвести контроль целостности информации путём подсчёта контрольных сумм на стадии Окончание Блочной SDO Загрузки.

Параметр	Запрос/Индикация	Ответ/Подтверждение
Аргумент Номер SDO Мультиплексор Размер Проверка сумм Да Нет	**Обязателен** Обязателен Обязателен Необязателен Обязателен Как вариант Как вариант	
Удалённый результат Успех Проверка сумм Да Нет Размер_след _блока (опред. принимающей стороной)		**Обязателен** Обязателен Обязателен Как вариант Как вариант Обязателен

Таблица 13. Инициациация Блочной SDO загрузки.

Это квитируемый сервис. В ответе сервер сообщает клиенту (удалённый результат – всегда успех) максимальное количество сегментов в начинаемом блоке, а так же способность/требование проверять контрольные суммы в конце блока. В случае отказа сервер должен отправить клиенту сообщение Аварийное завершение SDO трансферта.

9.2.2.1.10. Загрузка SDO блока.

При помощи этого сервиса клиент SDO осуществляет загрузку данных загружаемого блока на сервер. Блок данных передаётся как последовательность сегментов. Каждый сегмент содержит данные и порядковый номер начиная с 1, и до Размер_след _блока заданного на этапе инициирования трансферта, этот параметр может

быть пересмотрен сервером(в сторону уменьшения??) на этапе подтверждения приёма блока. Параметр продолжения указывает серверу оставаться в сосстоянии приёма блока или переходить к процедуре окончания приёма блока.

Параметр	Запрос/Индикация	Ответ/Подтверждение
Аргумент Номер SDO Данные Параметр продолжения Загрузка не завершена Загрузка завершена	**Обязателен** Обязателен Обязателен Обязателен Как вариант Как вариант	
Удалённый результат Успех Последний проквитированный сегмент Размер_след_блока.		**Обязателен** Обязателен Обязателен Обязателен

Таблица 14. Блочная SDO загрузка.

Это квитируемый сервис. Параметр удалённого результата – всегда успех.

В случае невозможности продолжать трансферт, вмекто квитирования должен быть вызван сервис Аварийное завершение SDO трансферта.

В случае нормального завершения сервер сообщает номер последнего успешно принятого сегмента. Если количество переданных клиентом и закачанных сервером байтов не совпадает, то клиент следующий блок начнёт с ретрансляции всех непроквитированных сервером сегментов. После квитирования блока сервер поместил всю принятую информацию по месту назначения и готов к приёму следующего блока информации размером не более Размер_след_блока. Может быть не более одного активного Блок-трансферта(в сети?? на сервере??). Сервису Блочная SDO загрузка должно предшествовать удачное выполнение сервиса Инициация блочной SDO загрузки.

9.2.2.1.11. Окончание Блочной SDO загрузки.

Этим сервисом завершается Блочная SDO загрузка. Для сервера указывается сколько байт в последнем сегменте не содержали информацию и шли порожняком (Значащая_информация).

Если на этапе Инициация Блочной SDO загрузки и сервер и клиент показали возможность и необходимость подсчёта контрольных сумм, то в этом сервисном примитиве клиент сообщает серверу подсчитанную им контрольную сумму.

Параметр	Запрос/Индикация	Ответ/Подтверждение
Аргумент Номер SDO Значащая_информация Контрольная сумма	**Обязателен** Обязателен Обязателен Обязателен, если была договорённость на этапе инициации	
Удалённый результат Успех		**Обязателен** Обязателен

Таблица 15. Окончание Блочной SDO загрузки.

Это подтверждаемый сервис. Удалённый параметр результата всегда показывает успех, если оговаривалась проверка контрольных сумм то при совпадении. В случае отказа(в т.ч. и несовпадении ChSym) должен быть выполнен сервис Аварийное завершение SDO трансферта.

9.2.2.1.12. Блочная SDO выгрузка(upload).

При помощи этого сервиса клиент SDO выгружает данные с SDO сервера(владельца Обьектного словаря), используя протокол Блочной SDO выгрузки.

Серверу указываются мультиплексор набора данных, которые должны быть выгружены и опционно размер.

Это квитируемый сервис. В случае отказа вместо подтверждения выполняется вызов сервиса Аварийное завершение SDO трансферта и опционно может быть указана причина отказа. В случае успеха клиент получает данные и опционно их размер.

Параметр	Запрос/Индикация	Ответ/Подтверждение
Аргумент Номер SDO Мультиплексор.	**Обязателен** Обязателен Обязателен	
Удалённый результат Успех Данные Размер Отказ Причина		**Обязателен** Как вариант Обязателен Необязателен Как вариант Необязателен

Таблица 16. Блочная SDO выгрузка.

9.2.2.1.13. Инициация Блочной SDO выгрузки.

При помощи этого сервиса клиент SDO запрашивает сервер SDO(владелец обьектного словаря) подготовится выгрузить данные для клиента.

Для сервера указывается мультиплексор запрашиваемого набора данных и количество данных, которые может принять клиент(размер блока в сегментах). Так же указывается граница переключения протоколов в байтах. Если количество байт предназначенных для скачивания меньше или равно этой границе, то сервер опционно может произвести трансферт этих данных по протоколу Выгрузка SDO, который описан в подпункте 9.2.2.1.4. И клиент и сервер могут указать возможность/необходимость проверки целостности передачи информации путём проверки контрольных сумм на стадии Окончание Блочной SDO выгрузки. Опционно клиенту может сообщаться фактический размер выгружаемой информации.

Параметр	Запрос/Индикация	Ответ/Подтверждение
Аргумент Номер SDO Размер_блока (опред. принимающей стороной) Подсчёт CRC Да Нет Мультиплексор Граница смены протоколов	**Обязателен** Обязателен Обязателен Обязателен Как вариант Как вариант Обязателен Обязателен	
Удалённый результат Успех Подсчёт CRC Да Нет Размер		**Обязателен** Обязателен Обязателен Как вариант Как вариант Необязателен

Таблица 17. Инициация SDO выгрузки.

Сервис квитируемый. В случае отказа сервером вместо подтверждения вызывается сервис Аварийное завершение SDO трансферта. В случае успеха в подтверждении для клиента опционно может быть указан размер выгружаемого блока данных. Если размер данных меньше или равен Границе переключения протоколов, то сервер может выгрузку по Протоколу Выгрузка SDO блока.

9.2.2.1.14. Выгрузка SDO блока.

При помощи этого сервиса клиент выгружает с SDO сервера очередной блок данных. Блок передаётся пользователю как последовательность сегментов. Каждый сегмент состоит из собственно данных, и номера в последовательности начиная с 1 и до Размер_блока, который оговаривается между клиентом и сервером на этапе Инициация SDO выгрузки и может меняться клиентом на этапе подтверждения блока. Параметр продолжения указывает клиенту должен ли он оставаться в фазе Загрузка Блока или перейти в фазу окончание блочной загрузки.

Параметр	Запрос/Индикация	Ответ/Подтверждение
Аргумент Номер SDO Данные Параметр продолжения Загрузка не завершена Загрузка завершена	**Обязателен** Обязателен Обязателен Обязателен Как вариант Как вариант	
Удалённый результат Успех Последний проквитированный сегмент Размер_след_блока.		**Обязателен** Обязателен Обязателен Обязателен

Таблица 18. Выгрузка SDO блока.

Этот сервис квитируемый. В случае удачной выгрузки клиент указывает номер последнего удачно принятого сегмента. Если номер не совпадает с номером последнего переданного сервером то сервер передаст все непроквитированные сегменты в следующем блоке. В случае невозможности дальнейшего проведения трансферта, должен быть выполнен сервис Аварийное завершение SDO трансферта. В случае успеха клиент перемещает все подтверждённые сегменты из буфера приёма и становится готов к приёму следующего блока. . Может быть не более одного активного Блок-трансферта(в сети?? на сервере??).

9.2.2.1.15. Окончание блочной SDO выгрузки.

Этим сервисом завершается Выгрузка SDO блока. Для клиента указывается количество пустых байт в последнем переданном сегменте. Если и клиент и сервер на этапе Инициации блочной SDO выгрузки показали необходимость проверки контрольных сумм, то сервер высылает контрольную сумму. Клиент должен посчитать контрольную сумму и сравнить с переданной сервером.

Параметр	Запрос/Индикация	Ответ/Подтверждение
Аргумент Номер SDO Значащая_информация Контрольная сумма	**Обязателен** Обязателен Обязателен Обязателен, если была договорённость на этапе инициации	
Удалённый результат Успех		**Обязателен** Обязателен

Таблица 19. Окончание Блочной SDO выгрузки.

9.2.2.2 SDO протоколы.

Для сервисного обьекта данных(SDO), производящего трансферт по сегментной или укороченной схеме определены шесть квитируемых сервисов(сервисных примитивов):

SDO загрузка(download).
Инициация SDO загрузки.
Загрузка SDO Сегмента
SDO выгрузка(upload).
Инициация SDO выгрузки.
Выгрузка SDO Сегмента

и один неквитируемый сервис – Аварийное завершение SDO трансферта.

Для сервисного обьекта данных(SDO), производящего трансферт поблочно, определены восемь квитируемых сервисов(сервисных примитивов):

Блочная SDO загрузка(download).
Инициация блочной SDO загрузки.
Загрузка блока SDO.
Завершение блочной SDO загрузки
Блочная SDO выгрузка(upload).
Инициация блочной SDO выгрузки.
Выгрузка блока SDO.
Завершение блочной SDO выгрузки.

и один неквитируемый сервис – Аварийное завершение БлочногоSDO трансферта(?? нигде кроме как здесь не упоминаетмся).

9.2.2.2.1. Протоколы SDO загрузки.

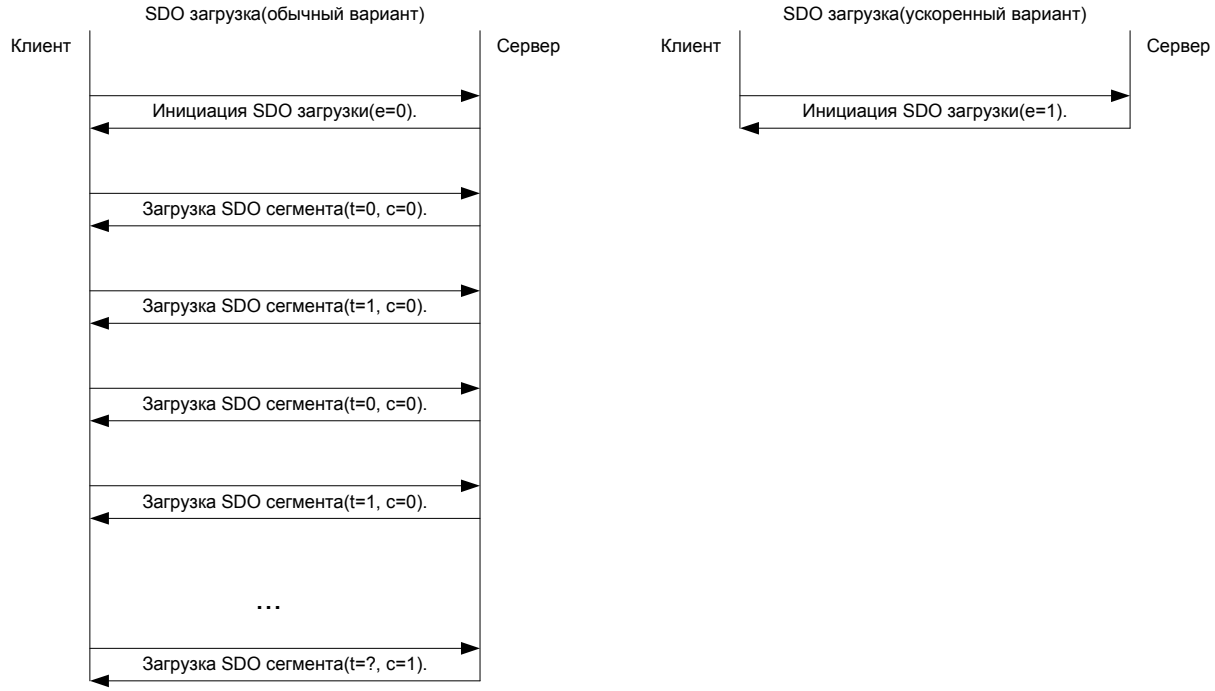


Рисунок 16. Протокол SDO загрузки.

Для осуществления загрузки SDO используется описываемый протокол. Загрузка данных происходит как последовательность вызовов сервиса Загрузка SDO сегмента, в количестве от нуля и более, начинаемая с вызова сервиса Инициация SDO загрузки. Последовательность прекращается в случае:

- на стадии Инициация SDO загрузки в запросе e – бит установлен в 1, при этом ответ сервера указывает на успешное завершение укороченного варианта SDO загрузки.
- на стадии Загрузка SDO сегмента, в ответе сервера установлен c – бит, указывающий на успешное завершение загрузки.
- в случае вызова сервиса Аварийное завершение SDO трансферта.
- в случае выполнения нового сервиса Инициация Доменной загрузки, который указывает на неуспешное завершение текущей последовательности загрузки и начала новой последовательности.

Если в двух последовательно принятых сегментах мерцающий бит не изменяется, содержимое последнего принятого сегмента должно быть проигнорировано. О такой ошибке сообщается приложению??, и оно может прервать загрузку.

9.2.2.2.2. Протокол Инициации SDO загрузки.

Данный протокол осуществляет Инициацию SDO загрузки.

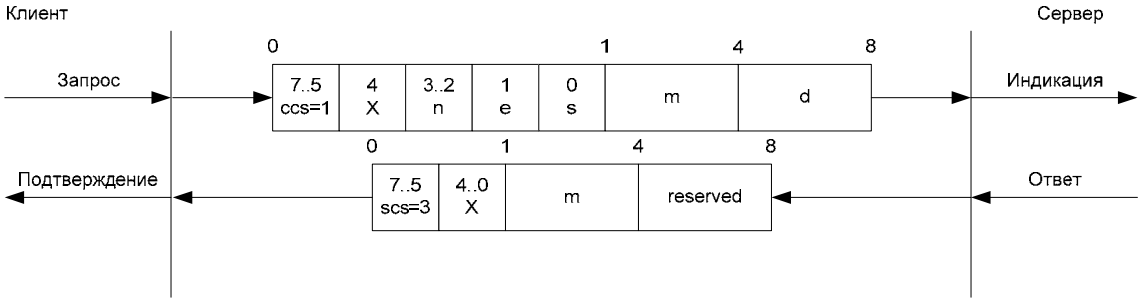


Рисунок 17. Протокол Инициации SDO загрузки.

- **ccs:** указатель команды клиента
  - 1 Запрос инициации загрузки.
- **scs:** указатель команды сервера
  - 3 Подтверждение инициации загрузки.
- **n:** имеет смысл только если e = 1 и s = 1, иначе 0. Показывает количество байт которые не содержат данные. Пустыми будут байты [8-n,7].
- **e:** тип трансферта
  - 0: Обычный трансферт
  - 1: Укороченный
- **s:** индикатор размера.
  - 0: Размер набора данных не указывается
  - 1: Размер набора данных указан.
- **m:** мультиплексор. Представляет Индекс/подиндекс набора данных, которые будут загружаться.
- **d:** данные

  e=0 s=0 поле d зарезервировано для будущих применений.

  e=0 s=1 поле d содержит количество байт которое нужно будет загрузить. Младший байт этого числа находится в байте 4, старший в байте 7.

  e=1 s=1 поле d содержит 4-n байт данных, кодировка которых зависит от типа данных, определяемого мультиплексором.

  e=1 s=0 поле d содержит неопределённое количество байтов информации
- **X:** не используется, всегда 0.
- **reserved:** зарезервировано для будущих применений, всегда 0.

9.2.2.2.3. Протокол Загрузка SDO сегмента.

Данный протокол осуществляет загрузку SDO сегмента.

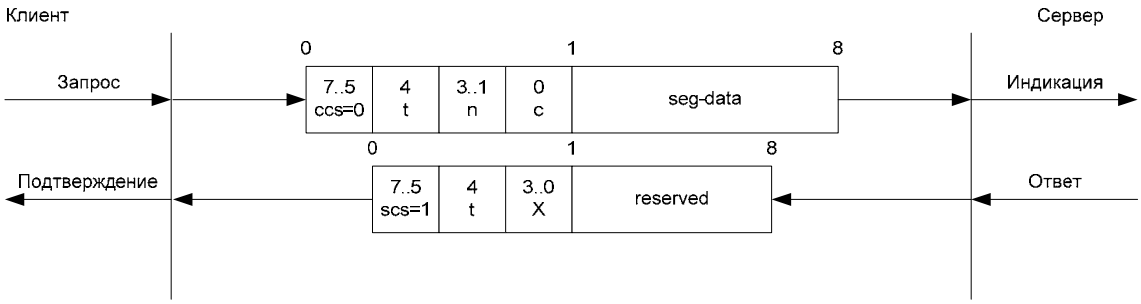


Рисунок 18. Протокол Загрузка SDO сегмента.

- **ccs:** указатель команды клиента
  - 0 Запрос загрузки сегмента.
- **scs:** указатель команды сервера
  - 3 Подтверждение загрузки сегмента
- **seg-data:** сегмент данных, может содержать до 7 байт данных кодировка которых зависит от типа данных, определяемого мультиплексором.
- **n:** показывает количество байт которые не содержат данные. Пустыми будут байты [8-n,7]. Если n=0, это означает что размер сегмента не указан.
- **c:** Параметр продолжения.
  - 0: Есть ещё сегменты для загрузки.
  - 1: Нет больше сегментов для загрузки.
- **m:** мультиплексор. Представляет Индекс/подиндекс набора данных, которые будут разгружаться.
- **t:** мерцающий бит. Должен меняться в каждом следующем сегменте последовательности, для первого сегмента в последовательности должен быть равен 0. Мерцающий бит в запросе и ответе должен совпадать.
- **X:** не используется, всегда 0.
- **reserved:** зарезервировано для будущих применений, всегда 0.

9.2.2.2.4. Протоколы SDO выгрузки.

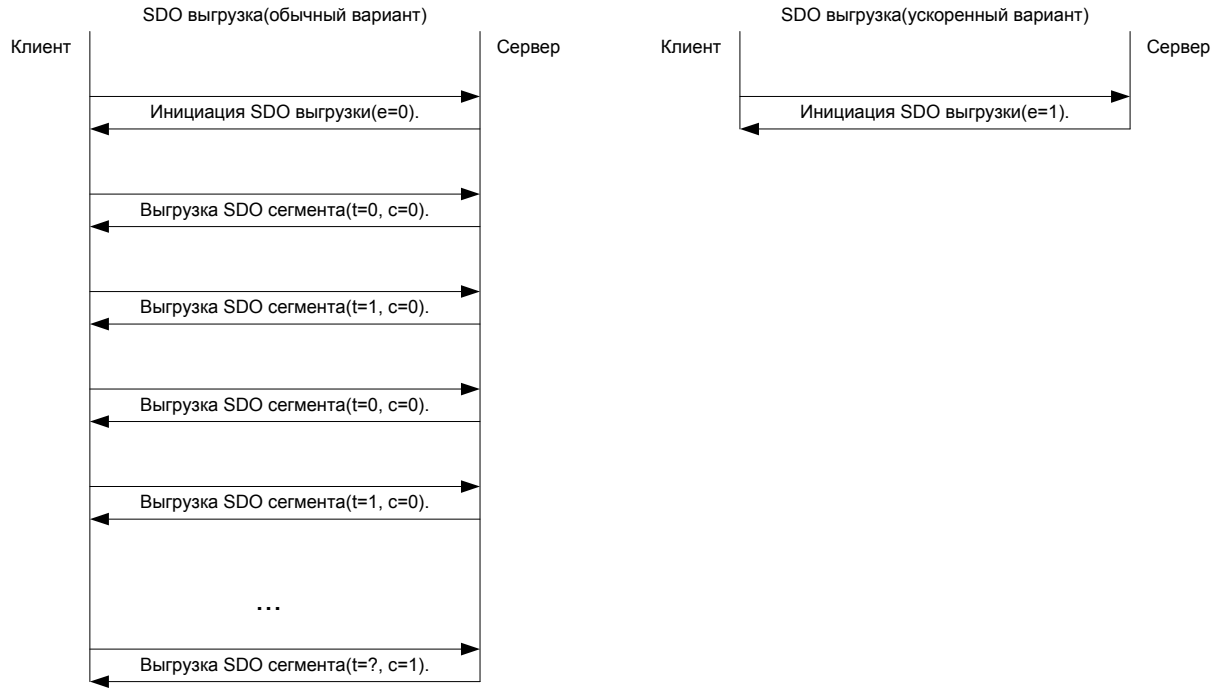


Рисунок 19. Протокол SDO выгрузки.

Для осуществления выгрузки SDO используется описываемый протокол. Выгрузка данных происходит как последовательность вызовов сервиса Выгрузка SDO сегмента, в количестве от нуля и более, начинаемая с вызова сервиса Инициация SDO выгрузки. Последовательность прекращается в случае:

- на стадии Инициация SDO выгрузки в ответе e – бит установлен в 1, при этом ответ сервера указывает на успешное завершение укороченного варианта SDO выгрузки.
- на стадии Выгрузка SDO сегмента, в ответе сервера установлен c – бит, указывающий на успешное завершение выгрузки.
- в случае вызова сервиса Аварийное завершение SDO трансферта.
- в случае выполнения нового сервиса Инициация SDO загрузки, который указывает на неуспешное завершение текущей последовательности загрузки и начала новой последовательности.

Если в двух последовательно принятых сегментах мерцающий бит не изменяется, содержимое последнего принятого сегмента должно быть проигнорировано. О такой ошибке сообщается приложению??, и оно может прервать загрузку.

9.2.2.2.5. Протокол Инициации SDO выгрузки.

Данный протокол осуществляет Инициацию SDO выгрузки.

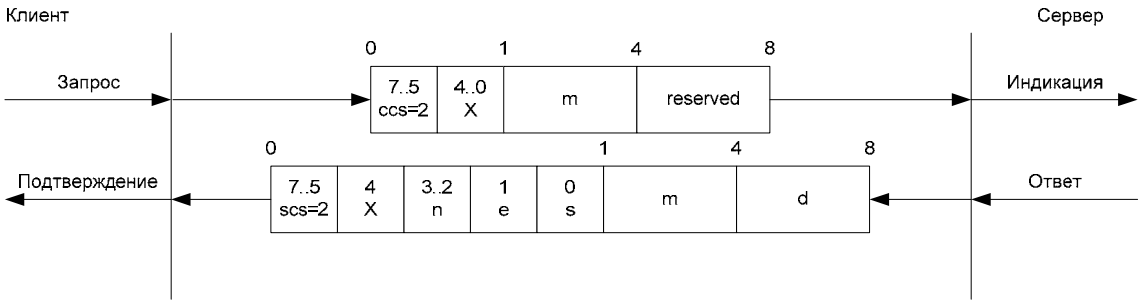


Рисунок 20. Протокол Инициации SDO выгрузки.

- **ccs:** указатель команды клиента
  - 2 Запрос инициации выгрузки.
- **scs:** указатель команды сервера
  - 2 Подтверждение инициации выгрузки.
- **n:** имеет смысл только если e = 1 и s = 1, иначе 0. Показывает количество байт которые не содержат данные. Пустыми будут байты [8-n,7].
- **e:** тип трансферта
  - 0: Обычный трансферт
  - 1: Укороченный
- **s:** индикатор размера.
  - 0: Размер набора данных не указывается
  - 1: Размер набора данных указан.
- **m:** мультиплексор. Представляет Индекс/подиндекс набора данных, которые будут выгружаться.
- **d:** данные

  e=0 s=0 поле d зарезервировано для будущих применений.

  e=0 s=1 поле d содержит количество байт которое нужно будет загрузить. Младший байт этого числа находится в байте 4, старший в байте 7.

  e=1 s=1 поле d содержит 4-n байт данных, кодировка которых зависит от типа данных, определяемого мультиплексором.

  e=1 s=0 поле d содержит неопределённое количество байтов информации
- **X:** не используется, всегда 0.
- **reserved:** зарезервировано для будущих применений, всегда 0.

9.2.2.2.6. Протокол Выгрузка SDO сегмента.

Данный протокол осуществляет выгрузку SDO сегмента.

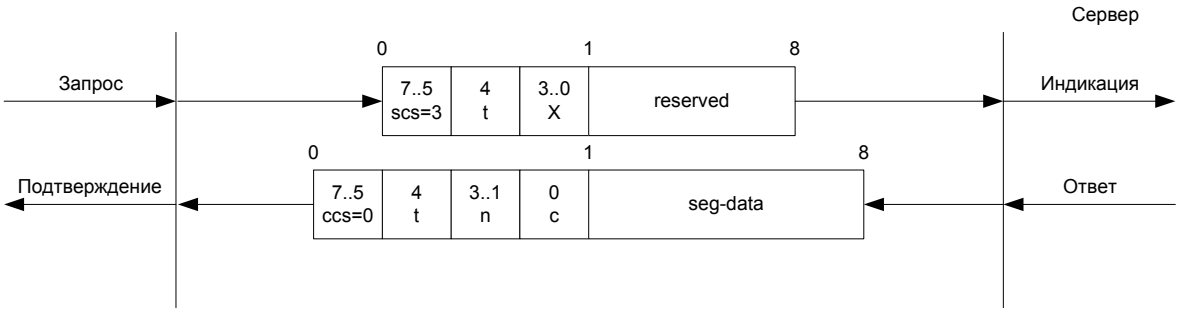


Рисунок 21. Протокол Выгрузка SDO сегмента.

- **ccs:** указатель команды клиента
  - 3 Запрос загрузки сегмента.
- **scs:** указатель команды сервера
  - 0 Подтверждение загрузки сегмента
- **seg-data:** сегмент данных, может содержать до 7 байт данных кодировка которых зависит от типа данных, определяемого мультиплексором.
- **n:** показывает количество байт которые не содержат данные. Пустыми будут байты [8-n,7]. Если n=0, это означает что размер сегмента не указан.
- **c:** Параметр продолжения.
  - 0: Есть ещё сегменты для выгрузки.
  - 1: Нет больше сегментов для выгрузки.
- **m:** мультиплексор. Представляет Индекс/подиндекс набора данных, которые будут разгружаться.
- **t:** мерцающий бит. Должен меняться в каждом следующем сегменте последовательности, для первого сегмента в последовательности должен быть равен 0. Мерцающий бит в запросе и ответе должен совпадать.
- **X:** не используется, всегда 0.
- **reserved:** зарезервировано для будущих применений, всегда 0.

9.2.2.2.7 Аварийное завершение SDO трансферта.

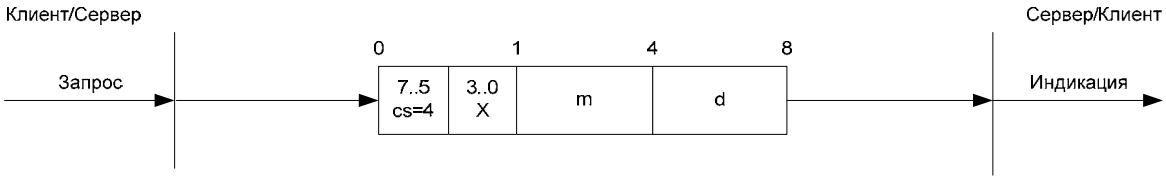


Рисунок 22. Протокол Аварийное завершение SDO трансферта.

- **cs:** указатель команды
  - 4 Запрос обрыва трансферта.
- **m:** мультиплексор. Представляет Индекс/подиндекс набора данных, которые будут разгружаться.
- **X:** не используется, всегда 0.
- **d:** содержит 4 байта сообщающие о причине отказа.

Код обрыва трансферта представляется как величина UNSIGNED32 .

Таблица 20. Коды аварийного завершения SDO трансферта.

Код завершения	Описание
0503 0000 h	Мерцающий бит не изменил своего состояния
0504 0000 h	Таймаут SDO протокола.
0504 0001 h	Указатель команды сервера/клиента содержит неверную или неизвестную команду
0504 0002 h	Неверный размер блока (только при блочном режиме).
0504 0003 h	Неверный порядковый номер сегмента.
0504 0004 h	Несовпадение контрольных сумм (только при блочном режиме).
0504 0005 h	Недостаточно памяти.
0601 0000 h	Неподдерживаемый доступ к обьекту.
0601 0001 h	Попытка считать обьект только для записи.
0601 0002 h	Попытка записать обьект только для чтения.
0602 0000 h	Несуществующий обьект словаря.
0604 0041 h	Обьект не может быть сопоставлен с PDO.
0604 0042 h	Количество и длинна обьектов, превышают длинну сопостовляемого с ними PDO.
0604 0043 h	Общая несовместимость параметра.
0604 0047 h	Общая внутренняя несовместимость внутри устройства.
0606 0000 h	Отказ в доступе из-за аппаратной ошибки.
0607 0010 h	Несоответствие типа данных, не совпадение длины сервисного обьекта.
0607 0012 h	Несоответствие типа данных, длина сервисного обьекта слишком велика.
0607 0013 h	Несоответствие типа данных, длина сервисного обьекта слишком мала.
0609 0011 h	Указаный подиндекс не существует.
0609 0030 h	Превышен диапазон значения параметра(только при записи параметра).
0609 0031 h	Значение записанного параметра слишком велико.
0609 0032 h	Значение записанного параметра слишком мало.

0609 0036 h	Максимальное значение меньше чем минимальное.
0800 0000 h	Общая ошибка.
0800 0020 h	Данные не могут быть переданы приложению, или запомнены им.
0800 0021 h	Данные не могут быть переданы приложению, или запомнены им, из-за запрета модулем локального контроля.
0800 0022 h	Данные не могут быть переданы приложению, или запомнены им, из-за имеющегося состояния устройства.
0800 0023 h	Ошибка динамического создания обьектного словаря или его отсутствие(например, если обьектный словарь генерируется из файла, и его содание не произведено из-за файловой ошибки).

Коды аварийного завершения не приведённые здесь зарезервированы.

9.2.2.2.8 Протокол блочной SDO загрузки.

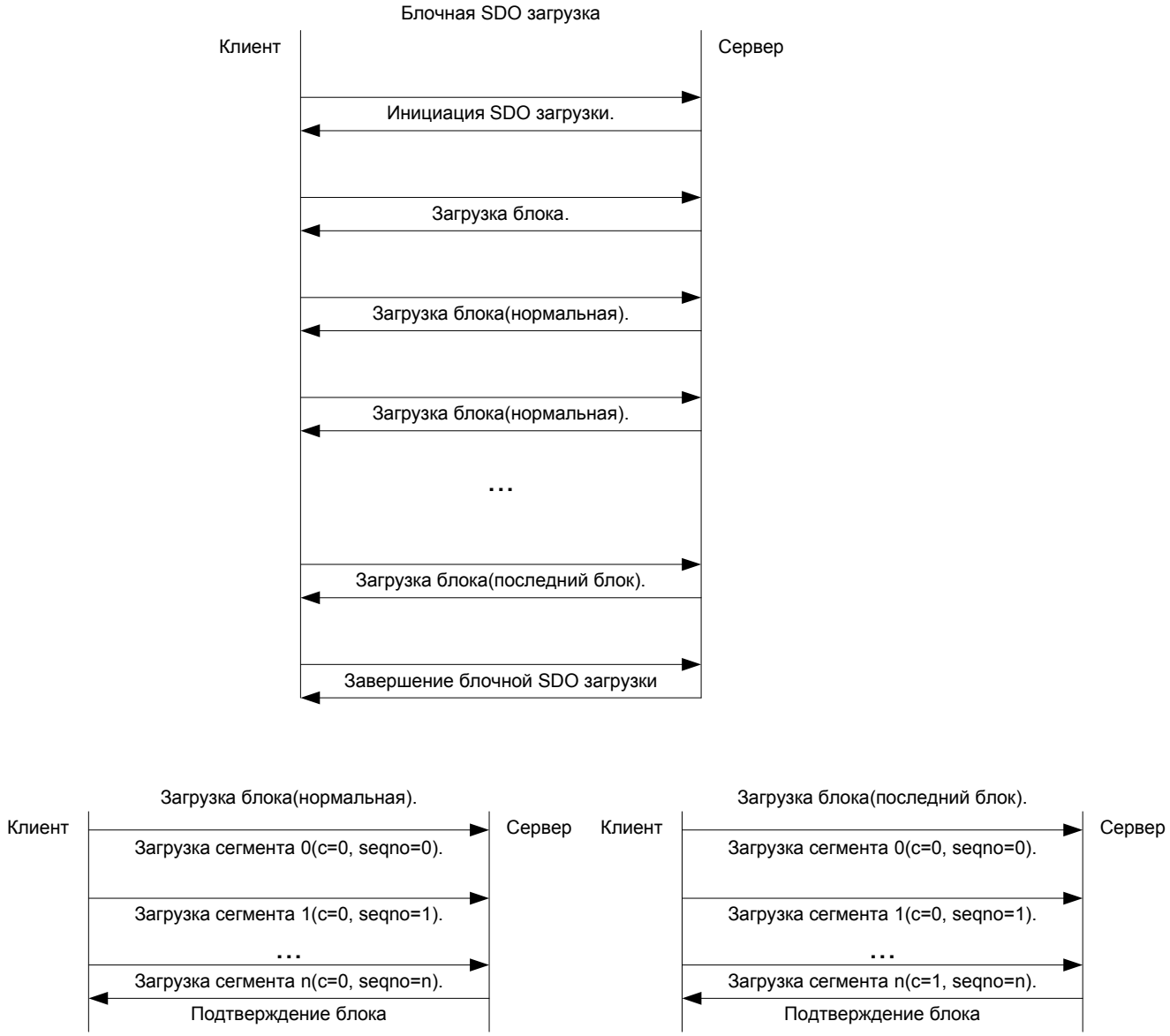


Рисунок 23. Протокол блочной SDO загрузки.

Данный протокол используется для осуществления блочной SDO загрузки. SDO в данном случае загружается как последовательность сервисов Загрузка SDO блока, с предшествующим ей сервисом Инициация блочной SDO загрузки. Блочная SDO загрузка прекращается в следующих случаях:

- Произошла загрузка сегмента блока с установленным c-битом, указывающим на завершение последовательности загружаемых блоков.
- Принят запрос/индикация Аварийное завершение SDO трансферта, указывающий на неуспешное завершение загрузки последовательности.

Весь сервис Блочная SDO загрузка завершается сервисом Завершение блочной SDO загрузки. Если на стадии сервиса Инициация блочной SDO загрузки клиент и сервер показали возможность генерировать CRC, то сервер должен посчитать CRC принятых данных. Если CRC, посчитанный сервером не совпадает с CRC, посчитанным клиентом, и сообщённым им серверу, то сервер должен сообщить об этом клиенту с помощью сервиса Аварийное завершение SDO трансферта.

9.2.2.2.9 Инициация блочной SDO загрузки.

По этому протоколу реализуется сервис Инициация блочной SDO загрузки.

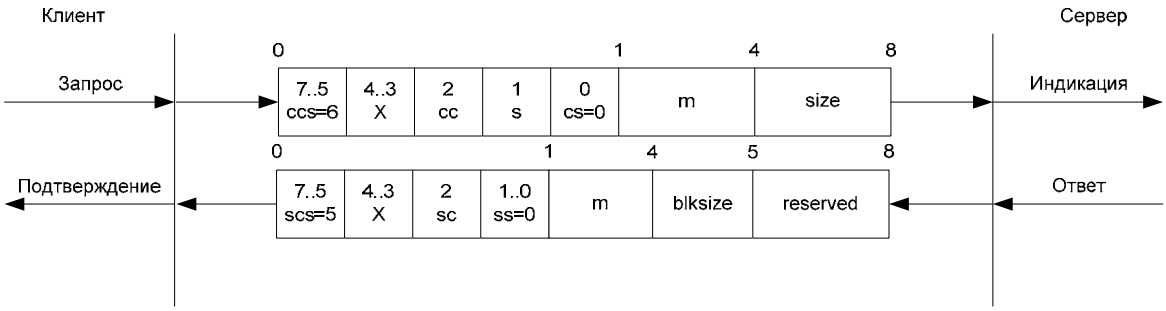


Рисунок 24. Инициация блочной SDO загрузки.

- **ccs:** указатель команды клиента
  - 6 Блочная загрузка.
- **scs:** указатель команды сервера
  - 2 Блочная загрузка.
- **s:** индикатор размера.
  - 0: Размер набора данных не указывается
  - 1: Размер набора данных указан.
- **cs:** подкоманда клиента.
  - 0: Запрос инициации загрузки.
- **ss:** подкоманда сервера.
  - 0: Подтверждение инициации загрузки.
- **cc:** поддержка CRC клиентом.
  - 0: Клиент не поддерживает генерацию CRC данных.
  - 1: Клиент поддерживает генерацию CRC данных.
- **cs:** поддержка CRC сервером.
  - 0: Сервер не поддерживает генерацию CRC данных.
  - 1: Сервер поддерживает генерацию CRC данных.
- **m:** мультиплексор. Представляет Индекс/подиндекс набора данных, которые будут выгружаться.
- **size:** размер загружаемой информации в байтах.
  - s=0: size зарезервирован для будущих применений, всегда содержит 0.
  - s=1: size содержит количество байт для загрузки, lsb в байте 4, msb в байте 7.
- **blksize:** количество сегментов в блоке должно быть от 1 до 127, включительно.
- **X:** не используется, всегда 0.
- **reserved:** зарезервировано для будущих применений, всегда 0.

9.2.2.2.10 Протокол загрузки сегментов при блочной SDO загрузке.

По этому протоколу реализуется сервис блочной загрузки.

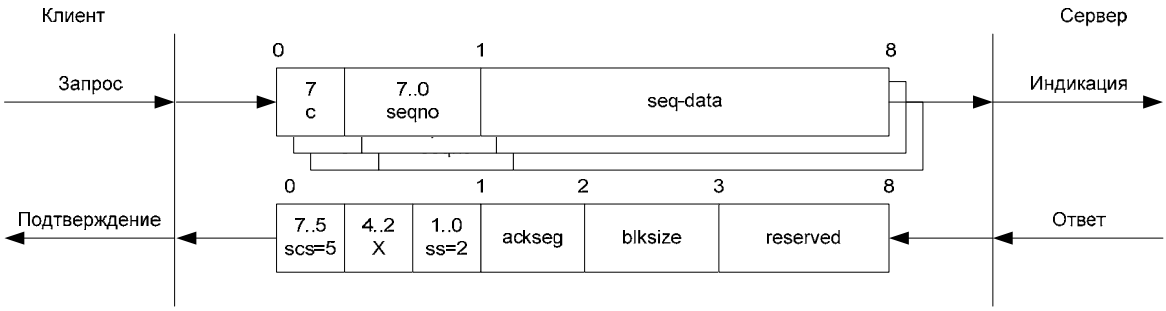


Рисунок 25. Протокол загрузки сегментов при блочной SDO загрузке.

- **scs:** указатель команды сервера
  - 2 Блочная загрузка.
- **ss:** подкоманда сервера.
  - 2: Подтверждение загрузки блока..
- **c:** показывает есть ли ещё сегменты для загрузки.
  - 0: есть сегменты для загрузки.
  - 1: нет больше сегментов для разгрузки, вход в фазу Завершение блочной SDO загрузки.
- **seqno:** порядковый номер сегмента 0<seqno<128.
- **seq-data:** до 7ми байт данных загружаемого сегмента.
- **ackseq:** порядковый номер последнего сегмента, который был принят успешно, во время передачи последнего блока. Если значение этого поля 0, то самый первый сегмент был принят неверно, и теперь необходимо передать заново всю последовательность сегментов последнего блока.
- **blksize:** количество сегментов в блоке, которое длжно быть принято клиентом для следующего блока, должно быть от 1 до 127, включительно.
- **X:** не используется, всегда 0.
- **reserved:** зарезервировано для будущих применений, всегда 0.

9.2.2.2.11 Протокол завершения блочной SDO загрузки.

По этому протоколу реализуется сервис блочной загрузки.

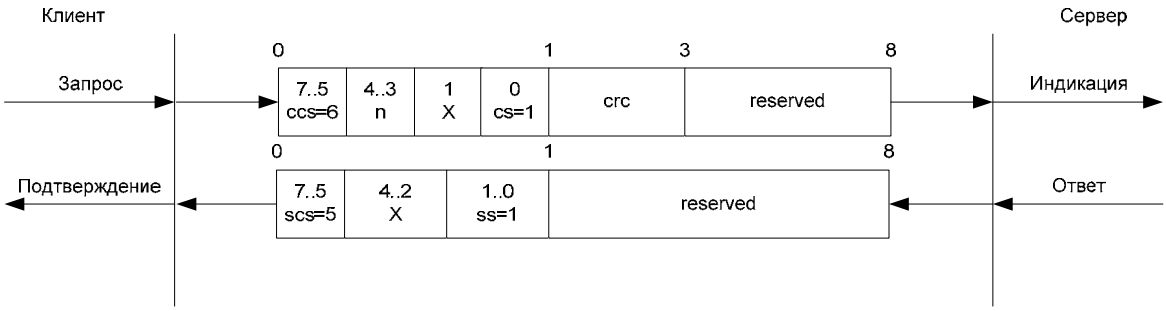


Рисунок 25. Протокол завершения блочной SDO загрузки.

- **ccs:** указатель команды клиента
  - 6 Блочная загрузка.
- **scs:** указатель команды сервера
  - 2 Блочная загрузка.
- **cs:** подкоманда клиента.
  - 1: Запрос завершения блочной SDO загрузки.
- **ss:** подкоманда сервера.
  - 1: завершения блочной SDO загрузки.
- **n:** Показывает сколько байт последнего сегмента последнего загруженного блока не содержат данных. Байты [8-n,7] будут пустыми.
- **CRC:** 16 бит Циклического Избыточного Кода (CRC), посчитанного для всего переданного набора данных. Алгоритм генерации CRC описан в п. 9.2.2.2.16. Данное поле актуально, только если на стадии Инициация блочной SDO загрузки биты cc и sc были установлены, иначе это поле должно содержать 0.
- **X:** не используется, всегда 0.
- **reserved:** зарезервировано для будущих применений, всегда 0.

9.2.2.2.11 Протокол блочной SDO выгрузки.

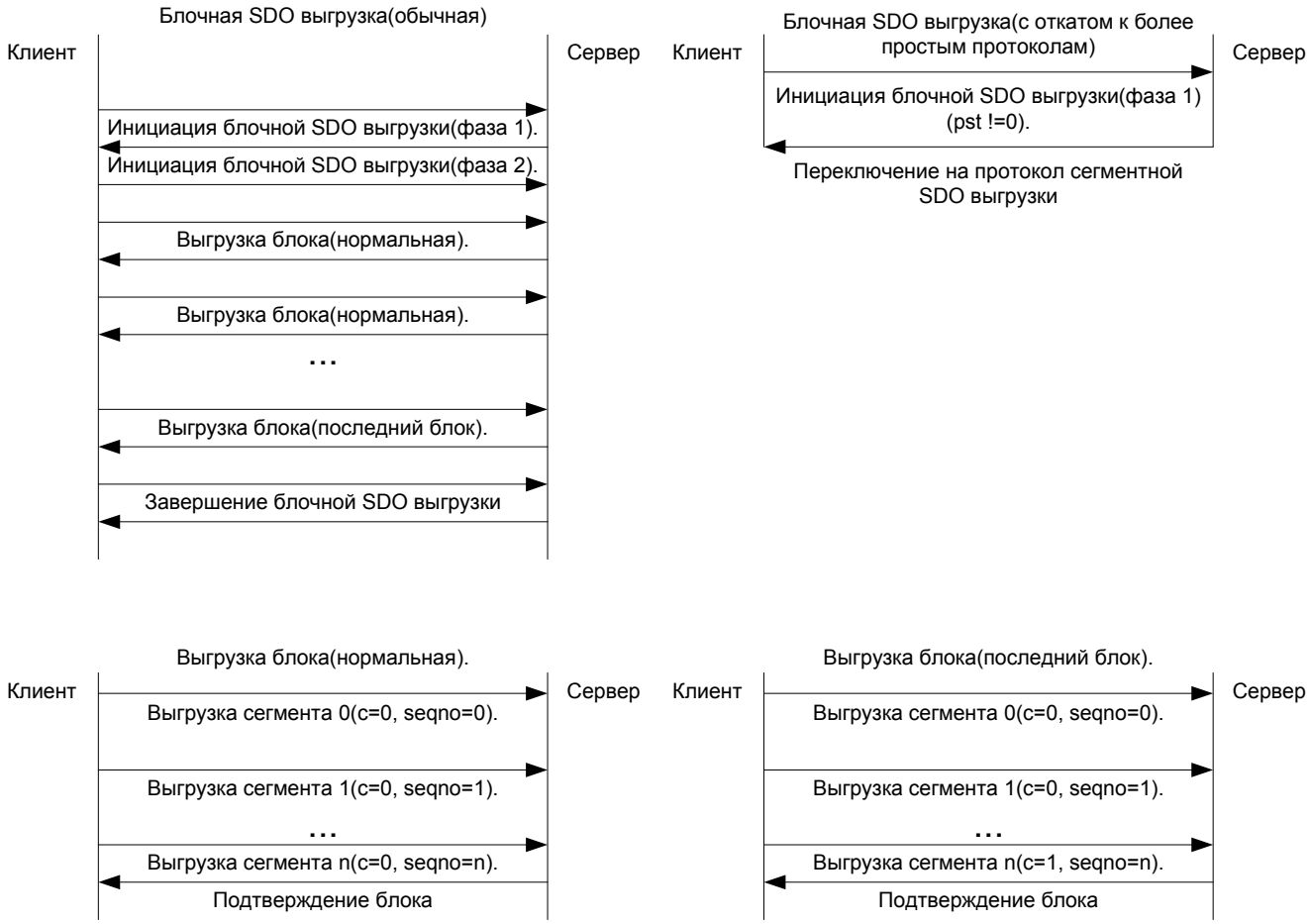


Рисунок 27. Протокол блочной SDO выгрузки.

По данному протоколу реализуется сервис Блочная SDO выгрузка, которая стартует с сервиса Инициация блочной SDO выгрузки. Клиент может указать серверу граничное значение размера блока данных в байтах, начиная с которого можно для увеличения производительности передачи использовать протокол Блочной SDO выгрузки, взамен протокола SDO выгрузки(обычной или укороченой). Если размер набора данных оказывается меньше либо равен значению этой границы, то сервер продолжет трансферт по протоколу обычной или укороченной передачи, иначе выгрузка SDO будет происходить как последовательность сервисов Выгрузка SDO блока. Данная последовательность прекращается в следующих случаях:

- Был выгружен сегмент с установленным с-битом, указывающим на завершение последовательности выгружаемых блоков.
- В результате запроса/индикации Аварийное завершение SDO трансферта, указывающего на неуспешное завершение выгрузки последовательности.

Весь сервис Блочная SDO выгрузка завершается сервисом Завершение блочной SDO выгрузки. Если клиент и сервер на стадии Инициация блочной SDO выгрузки указали свою способность подсчитывать CRC, то клиент должен подсчитать CRC всего объёма принятых данных, и сравнить с аналогичной величиной сообщённой ему сервером. Если эти величины не совпадают, то клиент вместо подтверждения должен послать серверу индикацию Аварийное завершение SDO трансферта.

9.2.2.2.13 Протокол Инициация блочной SDO выгрузки.

Данный протокол используется для реализации сервиса Инициация блочной SDO выгрузки. Если значение параметра pst(Protocol Switch Threshold), указанного клиентом в первом запросе превышает размер набора данных назначенного к выгрузке, то сервер может продолжить выгрузку по протоколу SDO выгрузка, который описан в п 9.2.2.2.4, для этого его ответ должен быть в формате описанном в п 9.2.2.2.5.

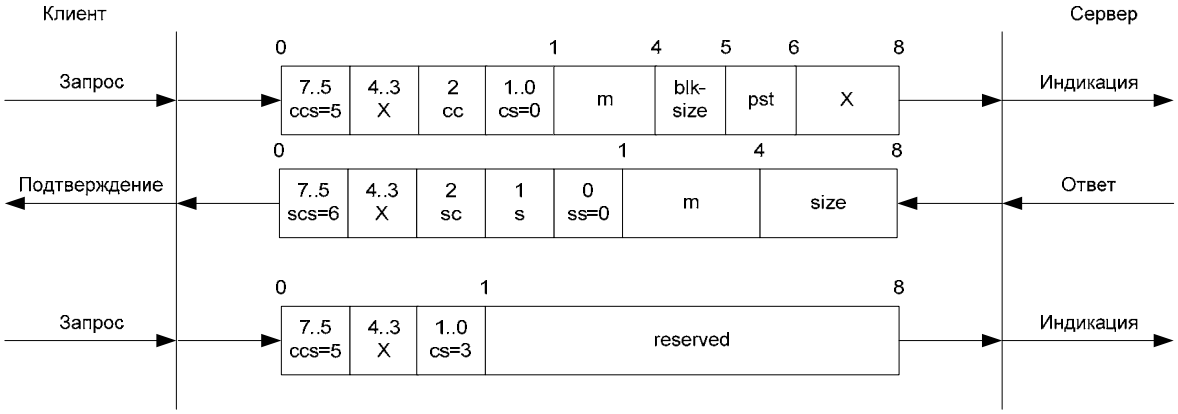


Рисунок 28. Протокол Инициация блочной SDO выгрузки.

- **ccs**: указатель команды клиента
  - 5: Блочная выгрузка.
- **scs**: указатель команды сервера
  - 6: Блочная выгрузка.
- **cs**: подкоманда клиента.
  - 0: Запрос инициации выгрузки.
  - 3: Старт выгрузки.
- **ss**: подкоманда сервера.
  - 0: Подтверждение инициации выгрузки.
- **m**: мультиплексор. Представляет Индекс/подиндекс набора данных, которые будут выгружаться.
- **cc**: поддержка CRC клиентом.
  - 0: Клиент не поддерживает генерацию CRC данных.
  - 1: Клиент поддерживает генерацию CRC данных.
- **cs**: поддержка CRC сервером.
  - 0: Сервер не поддерживает генерацию CRC данных.
  - 1: Сервер поддерживает генерацию CRC данных.
- **pst**: Граница Переключения Протоколов, в байтах для переключения протокола трансферта.
  - pst=0: Изменение протокола не допустимо.
  - pst>0: Если размер набора данных назначенного к выгрузке меньше или равен этой границе, то сервер может изменить протокол на протокол SDO выгрузка, передав ответ в формате этого протокола (п.9.2.2.2.5.)
- **s**: индикатор размера.
  - 0: Размер набора данных не указывается
  - 1: Размер набора данных указан.
- **size**: размер загружаемой информации в байтах.
  - s=0: size зарезервирован для будущих применений, всегда содержит 0.
  - s=1: size содержит количество байт для загрузки, lsb в байте 4, msb в байте 7.
- **blksize**: количество сегментов в блоке должно быть от 1 до 127, включительно.
- **X**: не используется, всегда 0.
- **reserved**: зарезервировано для будущих применений, всегда 0.

9.2.2.2.14 Протокол Выгрузка SDO блока.

Данный протокол используется для реализации сервиса Выгрузка SDO блока.

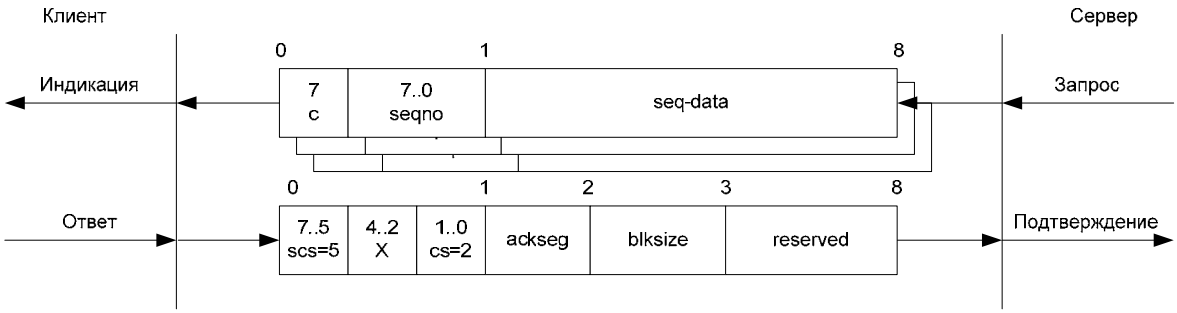


Рисунок 29. Протокол загрузки сегментов при блочной SDO загрузке.

- **ccs**: указатель команды клиента
  - 5: Блочная загрузка.
- **cs**: подкоманда клиента.
  - 2: Подтверждение загрузки блока..
- **c**: показывает есть ли ещё сегменты для выгрузки.
  - 0: есть сегменты для выгрузки.
  - 1: нет больше сегментов для выгрузки, вход в фазу Завершение блочной SDO выгрузки.
- **seqno**: порядковый номер сегмента 0<seqno<128.
- **seq-data**: до 7ми байт данных выгружаемого сегмента.
- **ackseq**: порядковый номер последнего сегмента, который был принят успешно, во время передачи последнего блока. Если значение этого поля 0, то самый первый сегмент был принят неверно, и теперь необходимо передать заново всю последовательность сегментов последнего блока.
- **blksize**: количество сегментов в блоке, которое длжно быть принято клиентом для следующего блока, должно быть от 1 до 127, включительно.
- **X**: не используется, всегда 0.
- **reserved**: зарезервировано для будущих применений, всегда 0.

9.2.2.2.15 Протокол завершения блочной SDO выгрузки.

По этому протоколу реализуется сервис блочной выгрузки.

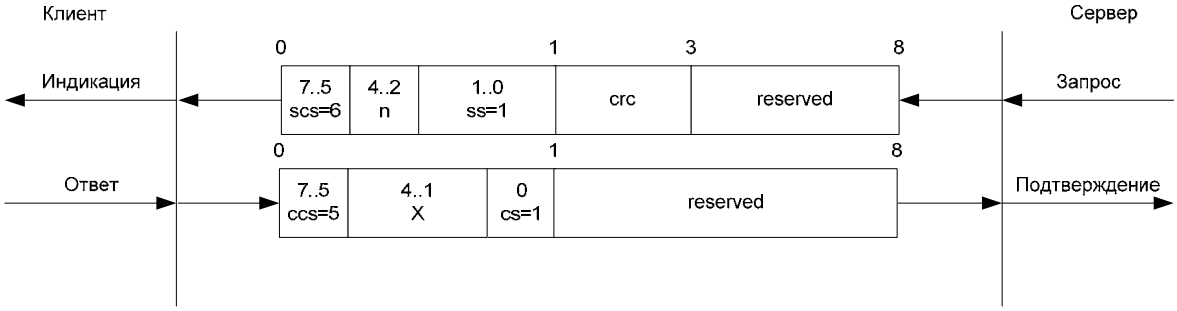


Рисунок 30. Протокол завершения блочной SDO выгрузки.

- **scs**: указатель команды сервера
  - 6: Блочная выгрузка.
- **ccs**: указатель команды клиента
  - 5: Блочная выгрузка.
- **ss**: подкоманда сервера.
  - 1: Запрос завершения блочной SDO выгрузки.
- **cs**: подкоманда клиента.
  - 1: Подтверждение завершения блочной SDO выгрузки.
- **n**: Показывает сколько байт последнего сегмента последнего выгруженного блока не содержат данных. Байты [8-n,7] будут пустыми.
- **CRC**: 16 бит Циклического Избыточного Кода (CRC), посчитанного для всего переданного набора данных. Алгоритм генерации CRC описан в п. 9.2.2.2.16. Данное поле актуально, только если на стадии Инициация блочной SDO выгрузки биты cc и sc были установлены, иначе это поле должно содержать 0.
- **X**: не используется, всегда 0.
- **reserved**: зарезервировано для будущих применений, всегда 0.

9.2.2.2.16 Алгоритм подсчёта контрольных сумм CRC, для проверки корректности блочного SDO трансферта .

Для проверки правильности блочного трансферта, клиент и сервер подсчитывают циклический избыточный код(aka контрольная сумма, aka CRC), которым обмениваются на стадии Завершение блочного SDO трансферта. Проверочный полином представлен в данном случае формулой `x^16 + x^12 + x^5 + 1`. Подсчёт производится с начальным значением 0.

9.2.3 Синхрообьекты (SYNC).

Синхрообьекты периодически широковещательно транслируются в сеть SYNCгенератором. SYNC-обьекты обеспечивают основное сетевое тактирование. Период времени между появлениями в сети двух последовательных SYNCов, определяется эталонным параметром период цикла связи(см. Обьект 1006h: Период Цикла Связи(Обьектный Словарь)), который может быть записан с помощью конфигурирующего инструментария приложением устройства, во время процесса начальной загрузки. Допускается некоторая задержка синхронизации источником синхрособщений, соответствующая приблизительно необходимому времени ожидания передачи сообщения, если источник непосредственно перед моментом синхронизации начал передачу таковую.

Для того чтобы гарантировать своевременный доступ синхрообьекта к шине CAN, он передаётся с очень высокоприоритетным идентификатором(1005h). Устройства которые работают синхронно, могут использовать SYNCи, для подстройки своей временной шкалы, со шкалой устройства-синхрогенератора. Тонкости такой подстройки являются устройство-зависимыми, поэтому их рассмотрение не попадает в рамки данного документа. Устройства которые требуют более точной всеобщей оси времени, могут использовать механизм высокоточной синхронизации описанный в п. 9.3.2.

9.2.3.1 SYNC-сервисы.

SYNCи передаются в соответствии с моделью взаимодействия производитель-потребитель, которая описана в п 6.3.3. Это не квитируемый сервис.
Атрибуты сервиса:

- тип пользователя: одно из значений `{consumer, producer}`.
- тип данных: `nil`(отсутствие)

9.2.3.1 SYNC-протокол.

Для данного обьекта определён один не квитируемый сервис Write SYNC.

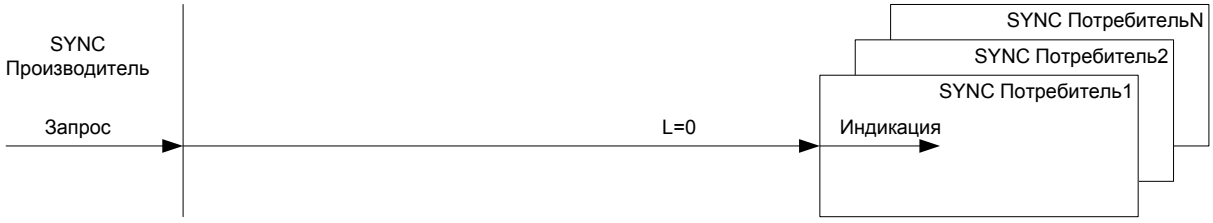


Рисунок 31. SYNC-протокол.

- SYNC не содержит каких либо данных(L=0), и следовательно не имеет полей.

Идентификатор SYNC-обьекта хранится в обьекте с индексом 1005h. ??

9.2.4. Обьект Отметка Времени (Time Stamp Object TIME).

Посредством обьекта Отметка Времени, устройствам сети предоставляется структура общего эталонного времени. Данный обьект содержит значение типа TIME_OF_DAY. Идентификатор TIME обьекта хранится в обьекте с индексом 1012h.

9.2.4.1 TIME сервисы.

TIMEsы передаются в соответствии с моделью взаимодействия производитель-потребитель, которая описана в п 6.3.3. Это не квитируемый сервис.
Атрибуты сервиса:

- тип пользователя: одно из значений `{consumer, producer}`.
- тип данных: `TIME_OF_DAY`.

9.2.4.1 TIME протокол.

Для данного обьекта определён один не квитируемый сервис Write SYNC.

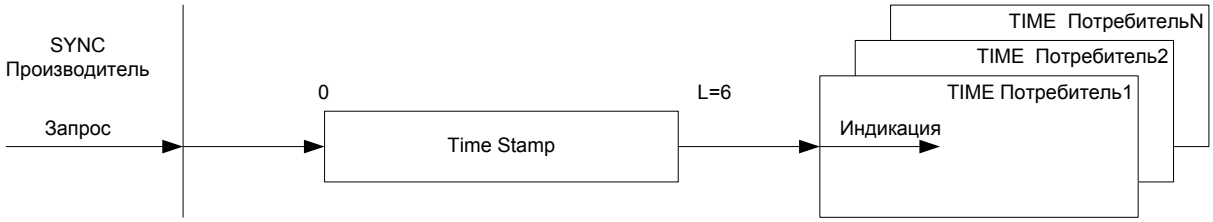


Рисунок 32. SYNC-протокол.

- **Time_stamp:** 6 байт данного обьекта(размер типа `TIME_OF_DAY`).

9.2.4. Авральные обьекты – срочные сообщения(Emergency Object - EMCY).

9.2.4.1 Применение срочных сообщений.

Передача срочных сообщений инициируется появлением внутренних ошибок устройства, их передачу осуществляет генератор срочных сообщений устройства. Такие обьекты подходят для аллертов об ошибках в виде прерывания. Авральные обьекты передаются в сеть только один раз за событие Ошибка, как бы долго это состояние не присутствовало. Только возникновение новых ошибок может инициировать передачу новых авральных обьектов. Каждое срочное сообщение могут принимать от нуля и более потребителей. Реакции потребителей срочных сообщений не оговариваются в рамках этого документа.

Данный документ определяет коды ошибок указанные в таблице 21 и структуру регистра ошибок, приведённую в таблице 48.

Код ошибки(hex)	Значение
00xx	Сброс ошибки.
10xx	Общая ошибка
20xx	Ток
21xx	Ток втекающий в устройство
22xx	Ток внутри устройства
23xx	Ток вытекающий из устройства
30xx	Напряжение
31xx	Питающее напряжение
32xx	Напряжение внутри устройства
33xx	Выходное напряжение
40xx	Температура
41xx	Температура окружающей среды
42xx	Температура устройства
50xx	«Железо» устройства
60xx	ПО устройства
61xx	Внутреннее ПО
62xx	ПО пользователя
63xx	Набор данных
70xx	Дополнительные модули
80xx	Мониторинг сети
81xx	Связь
8110	CAN перегружен(потеряны обьекты)
8120	CAN в очень плохом состоянии???
8130	Ошибка контроля существования узлов. Life Guard Error отсутствие запроса NMT мастера. Heartbeat Error отсутствие «сердцебиения»(т.е. контрольного тактирования)
8140	Восстановление узла после отсутствия CAN шины??
8150	Коллизия при передаче (совпадение COB-ID)
82xx	Ошибки протокола
8210	PDO не обработан по причине неверной длины обьекта
8220	Превышена длина PDO.
90xx	Внешняя ошибка.
f0xx	Дополнительная функция
ffxx	Определяется устройством

Таблица 21. Коды ошибок Авральных Обьектов.

Механизм срочных сообщений является необязательными для CANopen сети. Если устройство поддерживает данный механизм, то оно должно оперировать как минимум двумя кодами – 00xx и 10xx. Оперирование другими кодами необязательно.

Автомат аварийных состояний может находиться в двух состояниях(см рис 33). Передача срочных сообщений производится при переходах автомата в различные состояния. Связи между состояниями Автомата аварийных состояний и Автомата состояний NMT(управления сетью) определяются конфигурацией устройства.

0. После инициализации устройство входит в состояние Нет ошибок, если, конечно, они не были зафиксированы. Сообщение об отсутствии ошибок при этом не посылается.
1. Устройство обноружило внутреннюю ошибку, указываемую в первых трёх байтах срочного сообщения(структура сообщения показана в п. 9.2.5.2.). Автомат переходит в состояние Произошли ошибки. Это сопровождается отправкой аврального обьекта с соответствующим кодом ошибки и регистром ошибки. Код ошибки помещается в расположение обьекта 1003h(предопределённое поле ошибок).
2. Исчезла одна ошибка, но остались ещё ошибки. Может быть (должно??) отправлено срочное сообщение с аварийным кодом 0000h (Сброс ошибки), с указанием оставшихся ошибок, в регистре ошибок и в поле ошибок определяемом призводителем устройства.
3. Произошла новая ошибка. Автомат остаётся в состоянии ошибки, и передаёт срочное сообщение с соответствующим кодом ошибки. Новый код ошибки помещается на вершину массива кодов ошибок(1003h). Это гарантирует то что коды ошибок отсортированы в порядке возникновения(более старые с более-старшим индексом см. Обьект 1003h).
4. Все ошибки исчезли, автомат переходит в состояние Нет ошибок и передаёт срочное сообщение – «сброс ошибки/нет ошибок».

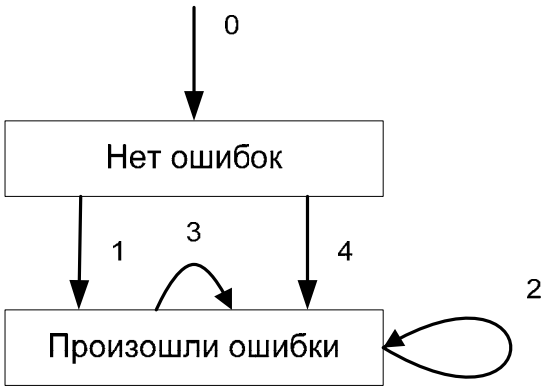


Рисунок 33. Диаграмма переходов Автомата аварийных состояний.

9.2.5.2. Структура срочных сообщений.

Телеграмма аврального обьекта состоит из 8 байт, как это показано на рисунке 34.

Байт	0	1	2	3	4	5	6	7
Содержимое	Код ошибки		Регистр ошибки	Поле ошибок, определённое изготовителем				

Рисунок 34. Данные срочного сообщения.

9.2.5.3. Сервисы срочных сообщений.

Срочные сообщения передаются, имея в основе Производитель-потребитель-Push модель взаимодействия, которая описана в п 6.3.3. Для срочного сообщения определены следующие атрибуты.

тип пользователя: уведомляющее устройство: `producer`.
принимающее устройство: `consumer`.

тип данных:
```
STRUCTURE OF
UNSIGNED(16) emergency_error_code,
UNSIGNED(8) error_register,
ARRAY (5) of UNSIGNED(8) manufacturer_specific_error_field
```

время подавления повторной передачи: определяется приложением.

9.2.5.4. Протокол срочных сообщений.

Определён один не квитируемый сервис - Write EMCY.

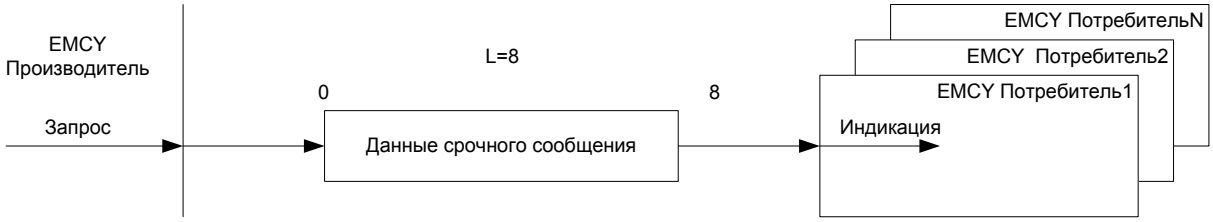


Рисунок 35. Протокол авральных обьектов - срочных сообщений.

Не допускается удалённый запрос передачи срочного сообщения.

9.2.6. Network Management Objects - Обьекты сетевого менеджмента.

Сетевой менеджмент(NMT) оперирует узлами(Node), и имеет в основе модель взаимодействия master-slave(п. 6.3.1.). NMT сервисы реализуются, обменом NMT обьектов между узлами. Через данные сервисы узлы инициализируются, запускаются, контролируются, сбрасываются и останавливаются. Все узлы рассматриваются как slaves. Каждый NMT slave однозначно идентифицируется в сети своим уникальным ID(Node-ID), который представляет число значением [1,127]. Сетевой менеджмент требует чтобы одно устройство в сети исполняло функции мастера.

9.2.6.1 NMTсервисы.

9.2.6.1.1 Module Control Services - Сервисы управления модулями.

При помощи данных сервисов NMT мастер управляет состоянием NMT слейвов. Атрибут состояния одно из значений {STOPPED, PRE-OPERATIONAL, OPERATIONAL, INITIALISING}. Сервисы контроля модулей могут быть выполнены как для отдельно взятого узла, так и для всех узлов сети одновременно. NMT-мастер управляет своим NMT автоматом состояний при помощи локальных сервисов, которые зависят от реализации. Сервисы контроля модулей, исключая сервис Запуск удалённого узла, могут инициироваться локальным приложением.

Запуск удалённого узла.

При помощи этого сервиса NMT мастер переводит состояние выбранного NMT слейва в OPERATIONAL.

Параметр	Запрос/Индикация
Аргумент	**Обязателен**
Node-ID	Как вариант
All	Как вариант

Таблица 22. Запуск удалённого узла.

Данный сервис не квитируемый и обязательный для всех устройств с сетевым управлением. После выполнения данного сервиса состояние выбранных удалённых узлов должно стать OPERATIONAL.

Остановка удалённого узла.

При помощи этого сервиса NMT мастер переводит состояние выбранного NMT слейва(ов) в STOPPED.

Параметр	Запрос/Индикация
Аргумент	**Обязателен**
Node-ID	Как вариант
All	Как вариант

Таблица 23. Остановка удалённого узла.

Данный сервис не квитируемый и обязательный для всех устройств с сетевым управлением. После выполнения данного сервиса состояние выбранных удалённых узлов должно стать STOPPED.

Перевод удалённого узла в состояние готовности.

При помощи этого сервиса NMT мастер переводит состояние выбранного NMT слейва(ов) в PRE-OPERATIONAL.

Параметр	Запрос/Индикация
Аргумент	**Обязателен**
Node-ID	Как вариант
All	Как вариант

Таблица 24. Перевод удалённого узла в состояние готовности.

Данный сервис не квитируемый и обязательный для всех устройств с сетевым управлением. После выполнения данного сервиса состояние выбранных удалённых узлов должно стать PRE-OPERATIONAL.

Сброс узла.

При помощи этого сервиса NMT мастер переводит состояние выбранного NMT слейва(ов) из любого, в подсостояние «сброс приложения».

Параметр	Запрос/Индикация
Аргумент	**Обязателен**
Node-ID	Как вариант
All	Как вариант

Таблица 25. Перевод удалённого узла в состояние готовности.

Данный сервис не квитируемый и обязательный для всех устройств с сетевым управлением. После выполнения данного сервиса состояние выбранных удалённых узлов должно стать RESET APPLICATION.

Сброс устройства связи.

При помощи этого сервиса NMT мастер переводит состояние выбранного NMT слейва(ов) из любого, в подсостояние «Сброс устройства связи».

Параметр	Запрос/Индикация
Аргумент	**Обязателен**
Node-ID	Как вариант
All	Как вариант

Таблица 26. Перевод удалённого узла в состояние готовности.

Данный сервис не квитируемый и обязательный для всех устройств с сетевым управлением. После выполнения данного сервиса состояние выбранных удалённых узлов должно стать RESET COMMUNICATION.

9.2.6.1.2 Error Control Services - Сервисы контроля ошибок работы сети.

При помощи этих сервисов сетевой менеджмент определяет отказы в CAN-сетке. Например внутренние ошибки могут приводить к сбросу устройств в сети, или к изменению их состояния(сетевого статуса см п 9.2.6.1.1). Определение таких ошибок не входит в рамки этого документа.

Контроль ошибок достигается, в основном, периодической передачей устройством сообщений, при этом существует два схожих варианта контроля.

Протокол караула узлов(Node guarding protocol)- Контроль путём передачи NMT мастером сторожевых запросов каждому узлу. Если NMT slave не ответит в течении определённого промежутка времени(время сторожевого ожидания node life time), или, неожиданно, изменился его сетевой статус, то NMT мастер информирует своё сетевое управляющее приложение об этом событии(Node guarding event – ошибка караула узлов). В свою очередь, NMT slave может караулить наличие NMT мастера по приёму описанных выше сторожевых запросов (Life guarding), причём время сторожевого ожидания вычисляется перемножением известных из обьектного словаря величин - периода повторения сторожевых запросов(Node guard time) и коэффициент пропорциональности стороживого ожидания (lifetime factor). Если в течении вычисленного таким образом времени, NMT slave не был проконтролирован NMT мастером, то модуль контроля сети слейва информирует своё приложение об этом событии(Life Guarding Event отсутствие сторожевого запроса). Если период повторения сторожевых запросов(Node guard time) и коэффициент пропорциональности стороживого ожидания (lifetime factor) равны 0(это значение задаётся по умолчанию), то NMT slave не караулит наличие мастера (но при этом, он должен отвечать мастеру, чтобы тот мог караулить узлы). Процесс караула узлов начинается для слейва с приёма первого удалённого запроса мастера, это может произойти, начиная со стадии начальной загрузки, либо позднее.

Второй вариант контроля называется контрольное тактирование(heartbeat), и основывается на циклической передаче сообщений генератором контрольного тактирования(heartbeat producer). От одного и более устройств в сети осведомлены о наличии в сети таких сообщений. Если генератором(поставщиком) пропущен такт контрольного тактирования, локальные приложения всех устройств-потребителей контрольного тактирования (heartbeat consumer) будут проинформированы об этом событии(Heartbeat Event).

Для контроля узлов обязательно выполнение контроля либо по протоколу караула узлов, либо по протоколу контрольного тактирования.

Node guarding event - ошибка караула узла.

При помощи данного сервиса, NMT сервис предоставляет NMT мастеру сообщение о том что возникла или разрешилась удалённая ошибка в удалённом узле идентифицируемом NodeID.

Параметр	Запрос/Индикация
Аргумент	**Обязателен**
Node-ID	Обязателен
Состояние	Обязателен
Ошибка произошла	Как вариант
Ошибка разрешилась	Как вариант

Таблица 27. Node guarding event – ошибка караула узла.

Данный сервис инициируется мастером и является необязательным??

Life Guarding Event - отсутствие сторожевого запроса.

При помощи данного сервиса, NMT сервис предоставляет NMT слейву сообщение о том что возникла или разрешилась удалённая ошибка.

Параметр	Запрос/Индикация
Аргумент	**Обязателен**
Состояние	Обязателен
Ошибка произошла	Как вариант
Ошибка разрешилась	Как вариант

Таблица 28. Life Guarding Event - отсутствие сторожевого запроса.

Данный сервис инициируется слейвом и является необязательным??

Heartbeat Event - ошибка контрольного тактирования.

При помощи данного сервиса, потребители контролного тактирования сообщают о том что возникла или разрешилась ошибка контрольного тактирования, в узле идентифицируемом NodeID.

Параметр	Запрос/Индикация
Аргумент	**Обязателен**
Node-ID	Обязателен
Состояние	Обязателен
Ошибка произошла	Как вариант
Ошибка разрешилась	Как вариант

Таблица 29. Heartbeat Event - ошибка контрольного тактирования.

Данный сервис инициируется потребителем Heartbeat и является необязательным??

9.2.6.1.3 Bootup Service - Сервис начальной загрузки.

Bootup Event – окончание начальной загрузки.

При помощи этого сервиса NMT slave сообщает что его внутренний сетевой статус перешёл из состояния `INITIALISING` в состояние `PRE-OPERATIONAL`.

Параметр	Запрос/Индикация
Аргумент	**Обязателен**
нет	

Таблица 30. Bootup Event – окончание начальной загрузки.

Данный сервис инициируется слейвом и является необязательным??

9.2.6.2 NMT протоколы.

9.2.6.2.1 Module Control Protocols - Протоколы управления модулями.

Протокол Запуск удалённого узла.

Данный протокол используется для осуществления сервиса Запуск удалённого узла.

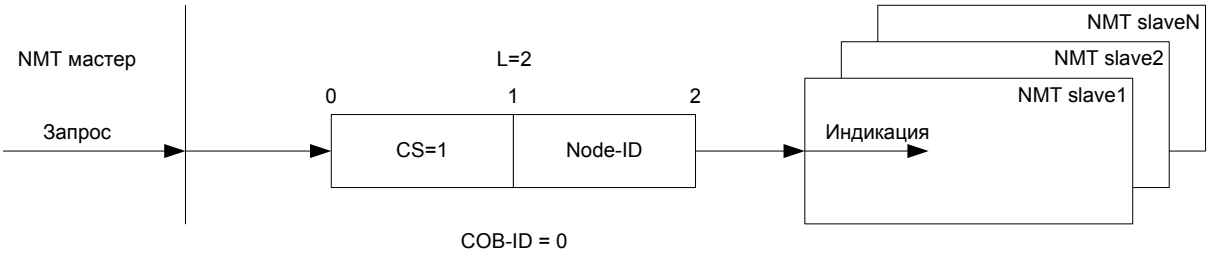


Рисунок 36. Протокол Запуск удалённого узла.

- **cs**: указатель команды NMT
  - 1: Старт.

Протокол Остановка удалённого узла.

Данный протокол используется для осуществления сервиса Остановка удалённого узла.

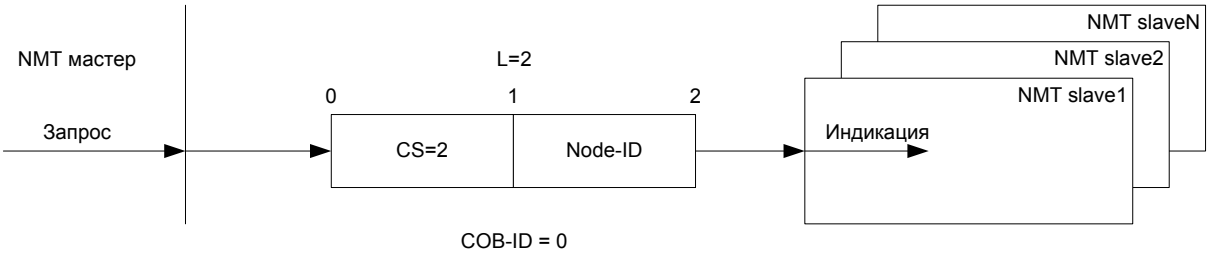


Рисунок 37. Протокол Остановка удалённого узла.

- **cs**: указатель команды NMT
  - 2: Стоп.

Протокол Перевод удалённого узла в состояние готовности.

Данный протокол используется для осуществления сервиса Перевод удалённого узла в состояние готовности.

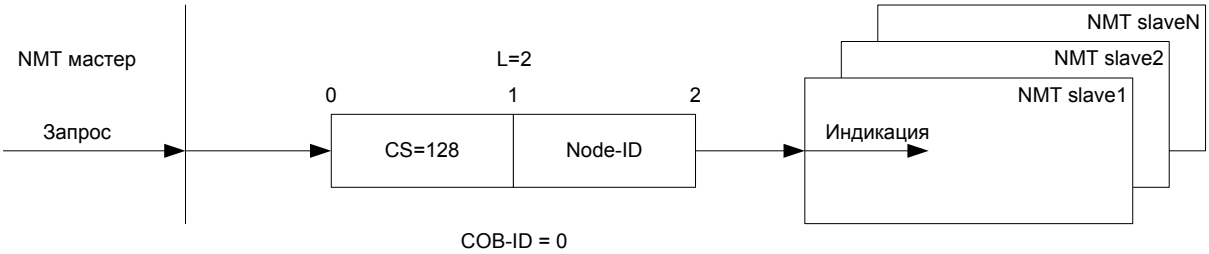


Рисунок 38. Протокол Перевод удалённого узла в состояние готовности.

- **cs**: указатель команды NMT
  - 128: Перевод удалённого узла в состояние готовности.

Протокол Сброс узла.

Данный протокол используется для осуществления сервиса Сброс узла.

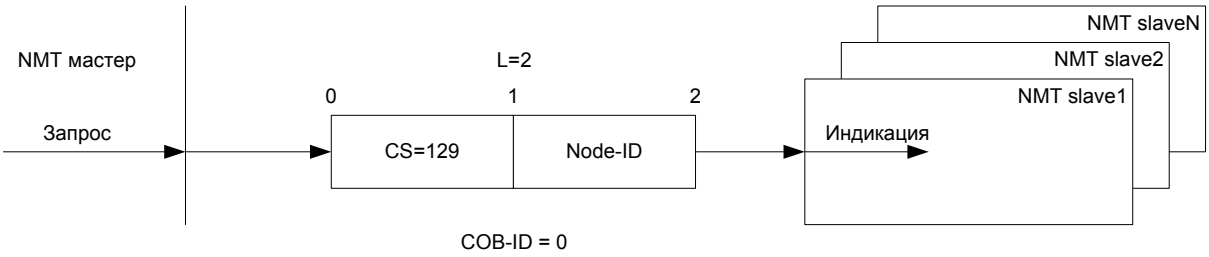


Рисунок 39. Протокол Сброс узла.

- **cs**: указатель команды NMT
  - 129: Сброс узла.

Протокол Сброс устройства связи.

Данный протокол используется для осуществления сервиса Сброс устройства связи.

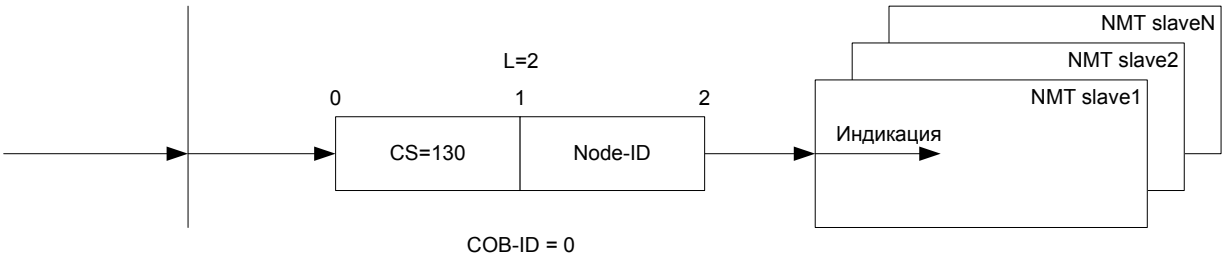


Рисунок 40. Протокол Сброс устройства связи.

- **cs**: указатель команды NMT
  - 130: Сброс устройства связи.

9.2.6.1.2 Протоколы контроля ошибок работы сети.

Протокол караула узлов(Node guarding protocol).

Данный протокол используется для обнаружения ошибок в сети. Каждому NMT слейву выделен под эти цели один коммуникационный обьект(COB). Этот протокол осуществляет инициируемый мастером сервис контроля ошибок.

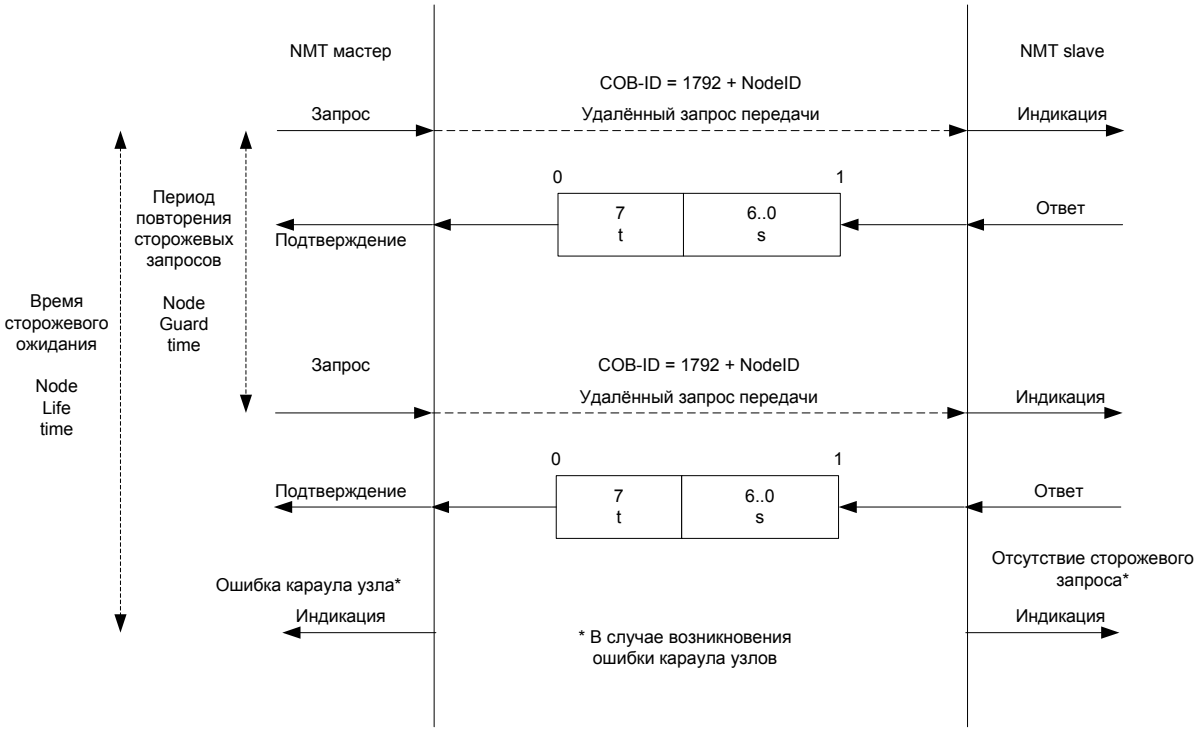


Рисунок 41. Протокол караула узлов.

- **s**: состояние NMT слейва.
  - 4: `STOPPED`
  - 5: `OPERATIONAL`
  - 127: `PRE-OPERATIONAL`
- **t**: мерцающий бит, значение этого бита должно меняться в двух последовательных ответах NMT слейва. Значение этого бита в первом ответе после активации данного протокола должно быть 0. Данный бит сбрасывается при выполнении сброса устройства связи, все остальные изменения сетевого статуса устройства не затрагивают этот бит. Если в принятом ответе NMT слейва указанный бит имеет то же значение что и в прошлом ответе, то реакция на такой ответ должна быть как будто ответ совсем не пришёл.

NMT мастер опрашивает каждого NMT слейва через равные промежутки времени. Данный временной интервал называется период повторения сторожевых запросов или сторожевой цикл(guard time), и может быть задан индивидуально для каждого узла в сети. Ответ слейва содержит сетевой статус этого устройства. Время сторожевого ожидания получается перемножением периода повторения сторожевых запросов и коэффициента пропорциональности стороживого ожидания (life time factor). Время сторожевого ожидания может быть задано индивидуально для каждого узла. Если слейв не был опрошен в течении этого времени его приложению сообщается об ошибке удалённого узла, при помощи сервиса Life Guarding Event - отсутствие сторожевого запроса.
Об ошибке удалённого узла мастера сообщается с помощью сервиса Node guarding event – ошибка караула узла, в следующих случаях:

- Удалённый запрос передачи не был подтверждён в течении времени сторожевого ожидания.
- Указанный в ответе сетевой статус слейва не соответствует ожидаемому.

Если было указано что произошла удалённая ошибка, а потом ошибка караула узлов исчезла, необходимо с помощью сервисов Node guarding event или Life Guarding Event сообщить приложению о том, что ошибка разрешилась.??
Для периода повторения сторожевых запросов и коэффициента пропорциональности стороживого ожидания задаются значения по умолчанию, хранящиеся в соответствующих элементах обьектного словаря.

Heartbeat Protcol - протокол контрольного тактирования.

Протокол контрольного тактирования предоставляет возможность контроля ошибок без посылки слейвами удалённых ответов. Устройство генератор контрольного тактирования периодически широковещательно рассылает обьекты-контрольные такты(heartbeats). От одного и более устройств сети получают их в виде индикации соответственно, такие устройства – потребители контрольного тактирования. Связи между генератором и потребителями контрольного тактирования конфигурируются посредством обьектного словаря. Потребители контролируют чтобы был приём контрольного такта-сообщения в течении времени ожидания контрольного тактирования(Heartbeat Consumer Time). Если в течении этого промежутка времени не происходит приёма контрольного такта, то генерируется событие - Heartbeat Event - ошибка контрольного тактирования.

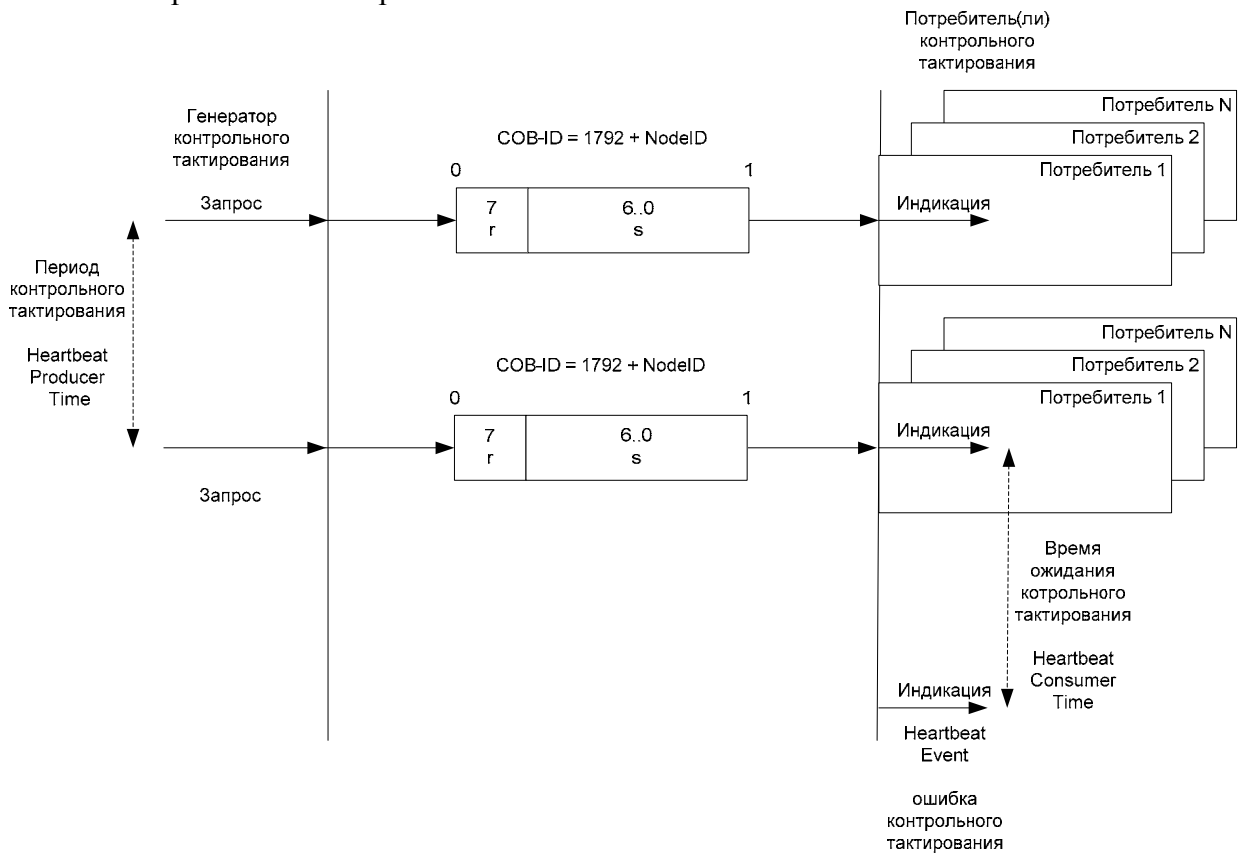


Рисунок 42. Протокол контрольного тактирования.

- **r**: зарезервирован(всегда 0).
- **s**: состояние устройства-генератора тактовых импульсов..
  - 0: `BOOTUP`
  - 4: `STOPPED`
  - 5: `OPERATIONAL`
  - 127: `PRE-OPERATIONAL`

Как только в устройстве сконфигурирован параметр период контрольного тактирования, немедлено включается протокол контрольного тактирования. Если устройство запускается с периодом контролного тактирования не равным 0, протокол контрольного тактирования запустится в момент перехода сетевого статуса устройства из состояния INITIALISING – инициализация в состояние PRE-OPERATIONAL – готовность. В таком случае первым таковым сообщением будет считаться Bootup сообщение.

Не допускается одновременное использование двух выше описанных механизмов контроля на одном устройстве(караул узлов и контрольное тактирование). Механизм контрольного тактирования обладает более высоким приоритетом, и если параметр период контрольного тактирования не равен 0, то запускается именно этот механизм.

Bootup Protocol - Протокол Окончание начальной загрузки.

Данный протокол используется для сигнализации о том что сетевой статус NMT слейва перешёл из состояния INITIALISING – инициализация в состояние PRE-OPERATIONAL – готовность. В этом протоколе используется тот же идентификатор COB-ID что и для протоколов контроля сетевых ошибок.

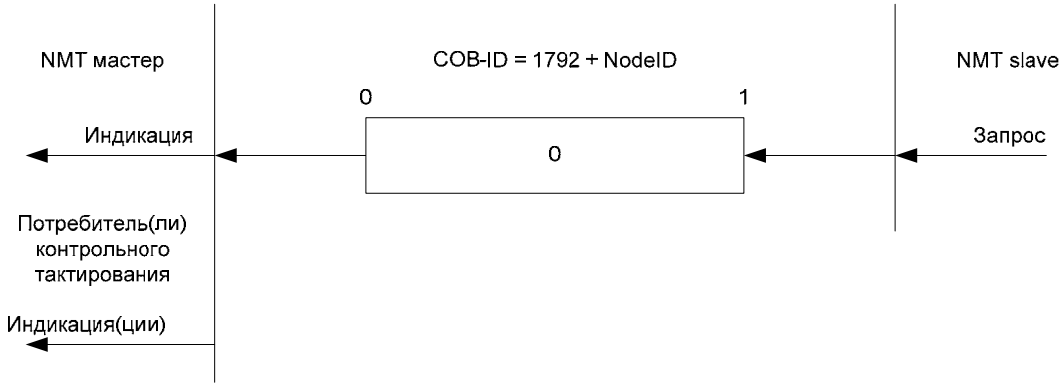


Рисунок 43. Протокол Окончание начальной загрузки

Передаётся один байт данных со значением 0.

9.3. Синхронизация потребителей SYNC обьектов.

9.3.1. Передача синхронных PDO сообщений.

Под синхронной передачей сообщений подразумевается то, что передача этих сообщений происходит в фиксированное время после передачи SYNC сообщений. Они передаются внутри заданного временного окна после SYNC передачи, самое большее один раз за период синхронизации.

В общем случае, фиксация времени передачи синхронных PDO относительно синхросообщений, наряду с периодичностью передачи синхросообщений, гарантирует, что устройства могут упорядоченно собирать переменные процесса из окружения процесса, и обновлять переменные координированно.(видимо буфер приёма.??). Устройства-потребители синхросообщений также могут снабжать синхронными PDO сеть. Приём синхронного сообщения означает момент, когда приложение взаимодействует с окружением процесса. Механизм синхронизации позволяет выполнять передачу управляющих воздействий, и обновление данных на фиксированной временной оси. В общем случае все синхронные PDO с управляющими воздействиями должны быть приняты до прихода синхросообщения. Только с приходом следующего за приёмом данных синхросообщения они будут обновлены для приложения. Так же приём

синхрособщения вызовет у устройства, работающего в цикличном режиме, сбор ответных данных и их передачу PDOsов с обновлёнными данными как можно скорее. В зависимости от возможностей устройства, оно может получать данные и не по синхрособщению, а в течении всего периода Окно синхронных PDO, в течении которого должны быть получены все управляющие воздействия. При этом оно может производить предварительную обработку управляющих воздействий которая потебуется для того чтобы обновить данные с приходом следующего синхрособщения.

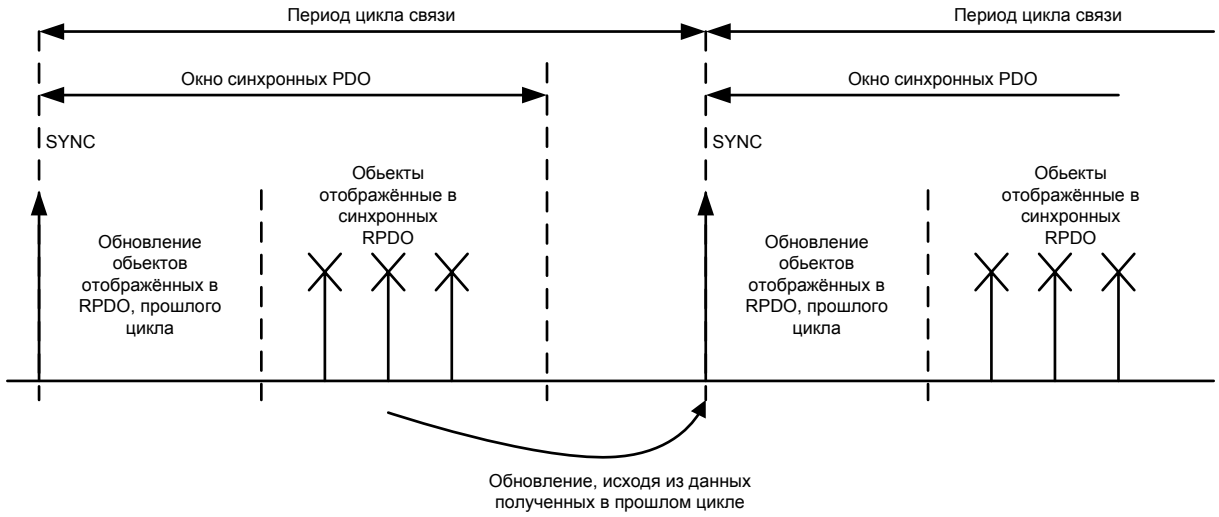


Рисунок 44. Синхронизация шины и обновление полученных данных.

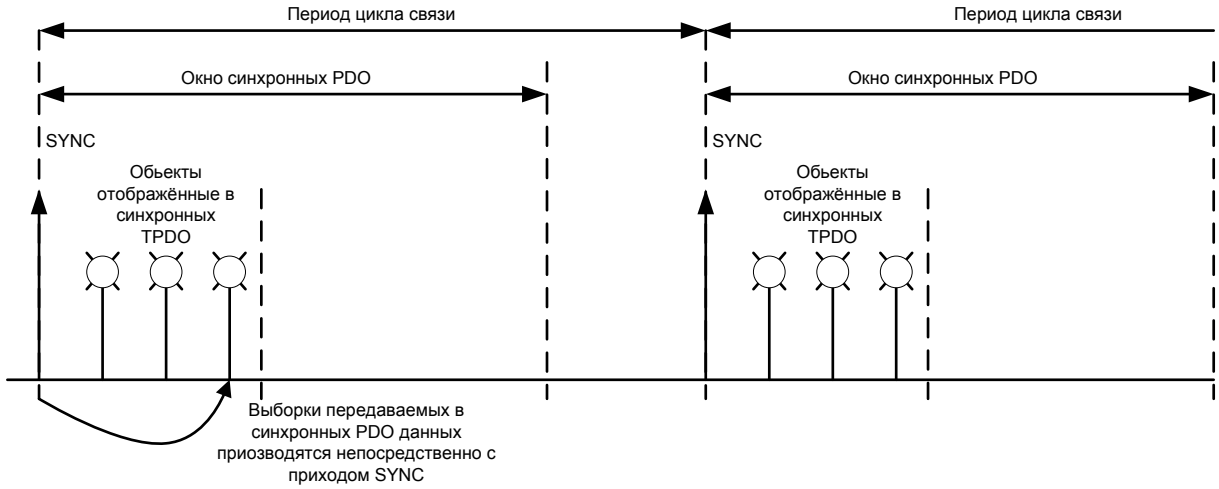


Рисунок 45. Синхронизация шины и передача собранных данных.

9.3.2. Дополнительный протокол высокоточной синхронизации.

Синхронизирующие сообщения не содержат данных, и поэтому легко синхронизируются. Однако, возможна задержка этих сообщений, так как даже очень высокоприоритетное сообщение должно дождаться окончания передачи текущего сообщения перед тем, как оно получит доступ к сети, и величина этой задержки зависит от битрейта сети.

Некоторые времякритичные приложения, особенно в больших сетях с уменьшеным битрейтом, требуют механизма более точной синхронизации, например, если необходимо синхронизировать внутренние часы с точностью до микросекунд. Это достигается при помощи дополнительного протокола высокоточной синхронизации, который использует

специальный вид сообщений отметка времени(см. рис. 46.), для коррекции неизбежного ухода локальных часов.

Суть механизма – в устройстве синхронизаторе сети происходит запрос прерывания, генерируемый в момент времени t1 окончанием успешной передачи синхросообщения, фактически прерывание происходит в момент времени t2. После этого в момент времени t4, он отправляет временную отметку с отметкой времени t1 (корректированное значением времени t2) - момент возникновения индикации об успешном завершении передачи. Потребитель синхросообщения отмечает при этом время t3, когда произошло его прерывание на событие, корректирует его и получает время t1 возникновения индикации об успешном завершении передачи, по локальным часам потребителя. Сравнивая эту величину с временной отметкой t1 полученной по сети, он имеет возможность произвести подстройку своих локальных часов. При использовании данного протокола только локальные временные задержки (t2-t1 для генератора и t3-t1 для потребителя) остаются узкими местами. Данные задержки обусловлены локальными явлениями(такими как время отклика на прерывание или аппаратные задержки) в отдельно взятом узле, и теоретически должны определяться эмпирическим путём. Точность определения локальных задержек определяется реализацией, и накладывает ограничение на точность высокоточной синхронизации. Заметим что каждый узел должен знать только своё собственное время задержки, так как генератором синхросообщений ему сообщается уже исправленная временная отметка(то есть не t2, а t1). Временная отметка представляется величиной типа `UNSIGNED32`, и содержит количество микросекунд, что означает что счётчик времени переполняется через каждые 72 часа. Этот сервис конфигурируется отображением высокоточной временной отметки внутри соответствующего PDO??.

Представляется логичным проводить подстройку часов только если максимальный уход локальных часов превышает точность синхронизации. Для большинства программно-схематехнических реализаций это означает, что достаточно сопровождать синхросообщение высокоточной отметкой времени один раз в секунду.

Такое решение позволяет получить лучшую точность синхронизации, которая может быть достигнута при подстройке через сеть, особенно когда CAN контроллеры поддерживают отметки времени(автоматически??). Заметим что точность такой настройки в большой степени независима от скорости передачи. Дальнейшее повышение точности требует добавочных схемотехнических решений.

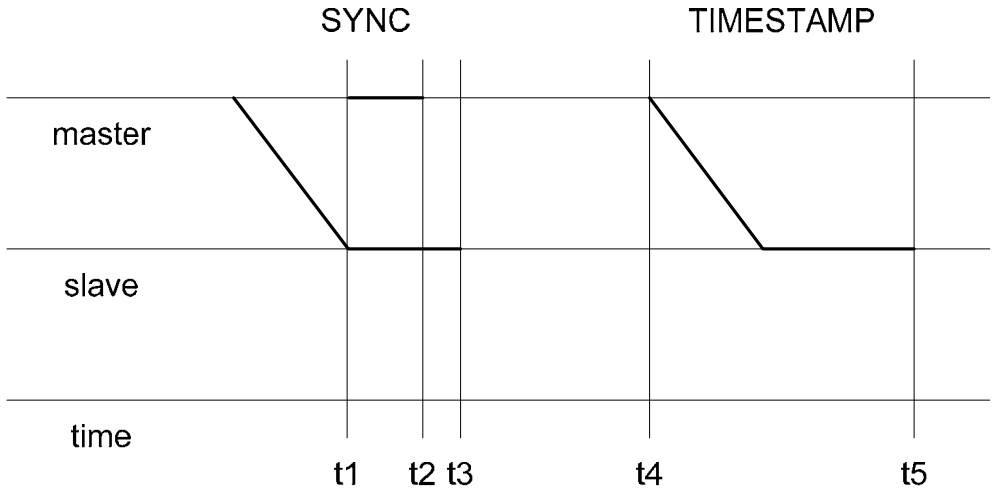


Рисунок 46. Дополнительный протокол высокоточной синхронизации.

9.4. Сетевая инициализация и системная начальная загрузка.

9.4.1. Процедура инициализации.

На рисунке 47 проказана общая структура сетевого инициализационного процесса, управляемого приложением NMT мастера, иначе приложением конфигурации.

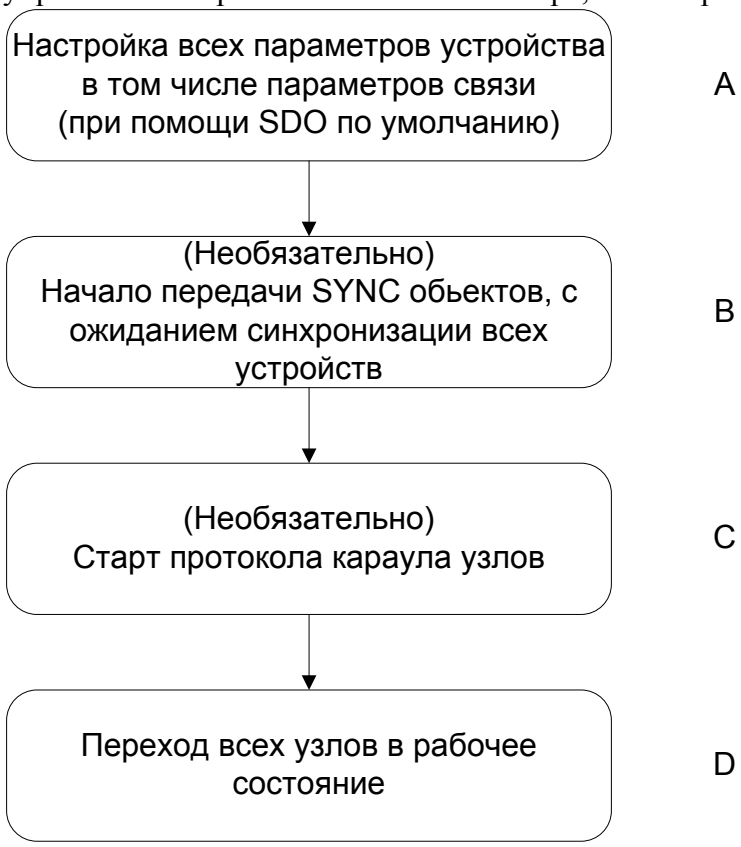


Рисунок 47. Блок-схема Сетевого инициализационного процесса.

На шаге A устройства имеют сетевой статус `PRE-OPERATIONAL`, в который они переходят самостоятельно после подачи питания. В этом состоянии доступ к устройствам возможен через их Предопределённый SDO обьект, через идентификаторы, которые заданы Предопределёнными Установками Связи. На данном шаге инициализации производится конфигурирование устройств(через сеть), для тех из них, которые поддерживают таковое. Это совершается Приложением или Инструментом конфигурирования которое постоянно располагается на клиентском узле для предопределённого SDO. Для устройств которые поддерживают эту особенность, допускается и/или конфигурирование PDOsов, отображающих прикладные обьекты (PDO-отображение), конфигурирование дополнительных SDO и, даже, установку COB-ID, всё описанное выполняется через предопределённый SDO. Во многих случаях конфигурация не требуется, так как по умолчанию заданы все переменные для приложения и параметры связи. Если приложение требует синхронизации всех или некоторых из узлов сети, то соответствующие механизмы могут быть запущены на необязательном шаге B, который призван гарантировать то, что все обьекты требующие синхронизации будут засинхронизированы, перед тем как получат сетевой статус `OPERATIONAL` на шаге D. Первая передача SYNC обьекта происходит спустя один цикл синхронизации после перехода узла в состояние `PRE-OPERATIONAL`. На шаге C запускаются, если поддерживаются, протоколы караула узлов используемые для контроля параметров сконфигурированных на шаге A. На шаге D всем узлам разрешается начать обмен данными посредством PDO.

9.4.2 Автомат состояний NMT.

9.4.2.1 Обзор.

На рисунке 48 показана диаграмма состояний абстрактного CANopen устройства. Устройства приобретают сетевой статус PRE-OPERATIONAL сразу после окончания инициализации устройства. Во время этого состояния возможна настройка параметров устройств и их ID через обьект сервисных данных(т.е. при помощи конфигурационного инструментария). После этого устройство может перейти непосредственно в состояние OPERATIONAL. Описываемый автомат определяет режим работы Модуля связи (см. рис 3 глава 6.2.). Связь между автоматами состояний - NMT и управляющего им (??) приложения попадают в рамки профилей устройств. Простейший протокол старта сети состоит из одного широковещательного сообщения - Start_remote_node.

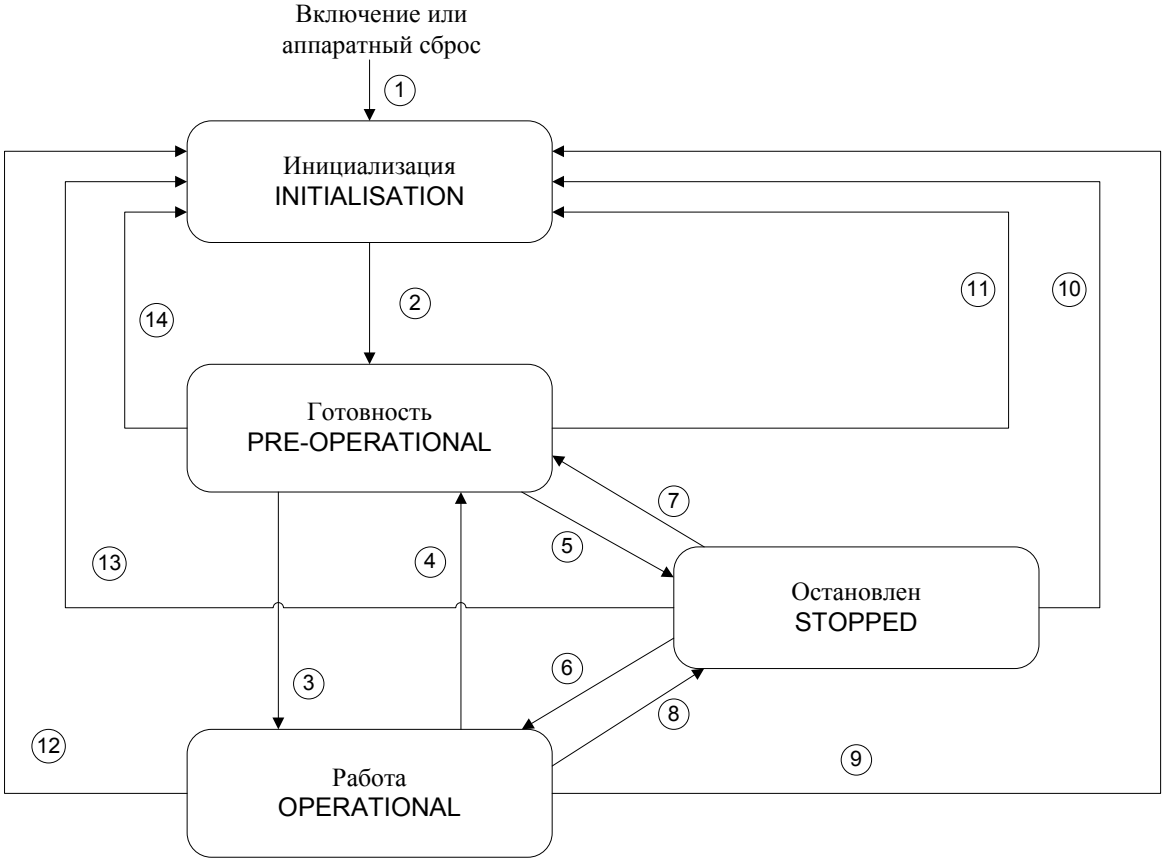


Рисунок 48. Диаграмма состояний абстрактного CANopen устройства.

Переход	Условие
1	После включения питания или аппаратного сброса автономно переходит в состояние INITIALISATION
2	Инициализация закончена – автоматически переход в состояние PRE-OPERATIONAL
3, 6	Индикация Start_Remote_Node(см 9.2.6.1.1)
4, 7	Индикация Enter_Pre-Operational (см 9.2.6.1.1)
5, 8	Индикация Stop_Remote_Node(см 9.2.6.1.1)
9, 10, 11	Индикация Reset_Node(см 9.2.6.1.1)
12, 13, 14	Индикация Reset_Communication(см 9.2.6.1.1)

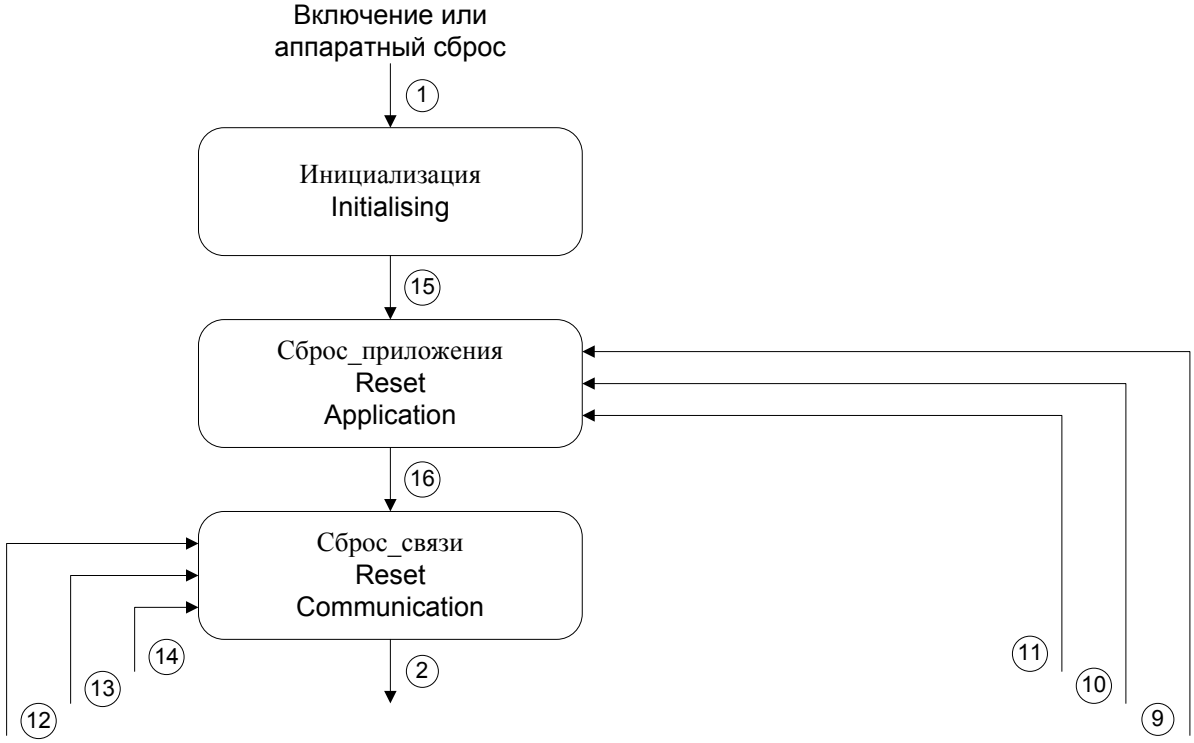
Таблица 31. Условия совершения переходов автоматом состояний.

9.4.2.2 Сетевые статусы aka состояния NMT автомата состояний.

9.4.2.2.1 Инициализация.

Состояеие «инициализация» разделяется на три подсостояния(см. рис. 49), расположенных в таком порядке, чтобы позволить осуществлять полный или частичный сброс узла.

1. **Инициализация**: Первое подсостояние, в него устройство переходит после подачи на него питания. После завершения основной инициализации узла устройство автономно(т.е. без получения команд через сеть) переходит в состояние Сброс_приложения.
2. **Сброс_приложения**: В этом состоянии принимают исходные значения(«по-включению») параметры области конфигурации специализированных устройств и области конфигурации стандартизованных устройств. После загрузки значений «по-включению», устройство автономно переходит в состояние Сброс_связи.
3. **Сброс_связи:** В этом состоянии принимают значения «по-включению» параметры области конфигурации связи. На этом состояние Инициализация завершается и устройство выполняет передачу Boot-Up сообщения и переходит в состояние **Готовность**.



Переход	Условие
1	После включения питания или аппаратного сброса автономно переходит в состояние `INITIALISATION`
2	Инициализация закончена – автоматически переход в состояние `PRE-OPERATIONAL`
5, 8	Индикация Stop_Remote_Node(см 9.2.6.1.1)
12, 13, 14	Индикация Reset_Communication(см 9.2.6.1.1)
9, 10, 11	Индикация Reset_Node(см 9.2.6.1.1)
15	Закончена первичная инициализация, автоматически в Сброс_приложения
16	Закончен Сброс_приложения, автоматически в Сброс_связи

Рисунок 49. Структура состояния Инициализация.

Значения «по-включению» это последние запомненные в энергонезависимой памяти значения. Если такое запоминание не поддерживается устройством, или не просто было выполнено, или сбросу предшествовала команда восстановить значения по умолчанию(обьект 1011h), то значениями «по включению» будут значения по умолчанию, описанные в профилях устройства и связи.

9.4.2.2.2 Готовность.

В данном состоянии возможна связь с использованием SDOsов. PDO не созданы, поэтому обмен ими не допустим. Конфигурирование PDOsов, параметров устройства и расположение обьектов приложений (PDO-отображение) могут быть выполнены при помощи конфигурирующего приложения. Данное состояние может быть переключено на состояние Работа посылкой узлу запроса Start_Remote_Node.

9.4.2.2.3 Работа.

В состоянии `OPERATIONAL` активны все коммуникационные обьекты. При переходе в это состояние создаются все PDO; конструктор использует для этого параметры описанные в Обьектном Словаре. Доступ к Обьектному Словарю возможен при помощи SDO. Однако, особенности реализации или автомат состояний приложения могут требовать ограничения на доступ к определённым обьектам, пока они задействованы в работе, например, обьект может содержать прикладную программу, которая не может быть изменена во время выполнения.

9.4.2.2.4 Остановлен.

Переключение устройства в данное состояние приводит к принудительной остановке связи в целом(исключая протокол караула узлов или контрольного тактирования, если активен). Помимо сказанного, данное состояние может быть использовано для установки определённого режима работы приложений. Описание данных загадочных режимов работы лежит в рамках описания профилей устройств.

9.4.2.3 Соответствие между сетевым статусом и разрешёнными для него обьектами связи.

	INITIALISING	PRE-OPERATIONAL	OPERATIONAL	STOPPED
PDO			X	
SDO		X	X	
Synchronisation Object		X	X	
Time Stamp Object		X	X	
Emergency Object		X	X	
Boot-Up Object	X			
Network Management Objects		X	X	X

Таблица 32. Соответствие между сетевым статусом и разрешёнными для него обьектами связи.

9.4.2.4 Изменение сетевого статуса.

Изменение сетевого статуса может вызываться следующими причинами:

- приём NMT обьектов, используемых в сервисах управления модулями.
- аппаратный сброс
- локально вызванные сервисы управления модулями, определённые в профилях устройств.

9.4.3 Предопределённые Установки Связи.

С целью уменьшения обьёма работ по конфигурации простых сетей, по умолчанию определена обязательная схема распределения идентификаторов Обьектного Словаря??. Эти идентификаторы доступны если устройство находится в состоянии Готовность, непосредственно после инициализации(если для устройства не были сохранены изменения этих идентификаторов). Обьекты SYNC, TIME STAMP, EMERGENCY и PDOsы могут быть удалены, или переопределены посредством динамического распределения. Устройство должно предоставлять соответствующие идентификаторы только для поддерживаемых обьектов связи. Структура определённго по умолчанию распределения ID (таблицы 33 и 34) содержит функциональную часть, определяющую приоритет обьектов, и Node-ID(идентификатор узла) часть, которая позволяет различать между собой устройства с одинаковой функциональностью. Это позволяет осуществлять соединения точка-точка между отдельно взятым мастером и количеством до 127 включительно, слейвами. Так же это поддерживает широковещательную неквитируемую передачу NMT-обьектов, SYNC и TIME STAMP обьектов. Широковещательность определяется указанием ID равным 0. Предопределённые Установки Связи поддерживают один авральный обьект, один SDO, до 4х передаваемых PDO(TPDO) и до 4х принимаемых PDO(RPDO) и NMT обьекты.

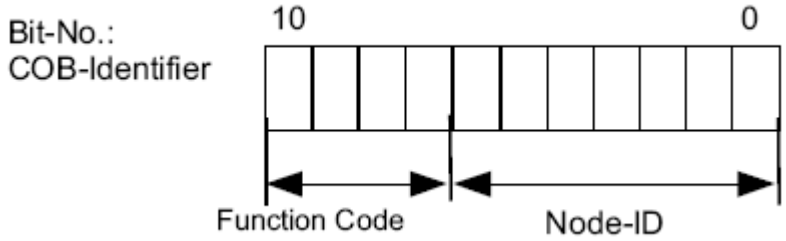


Рисунок 50. Структура распределения ID, определённого в Предопределённых Установках Связи.

Обьект	Функциональный код (двоичный)	Результирующий COB-ID	Индекс соответствующего параметра связи
NMT	0000	0	-
SYNC	0001	128(80h)	1005h,1006h,1007h
TIME STAMP	0010	256(100h)	1012h,1013h

Таблица 33. Широковещательные обьекты в Предопределённых Установках Связи.

Обьект	Функциональный код (двоичный)	Результирующий COB-ID	Индекс соответствующего параметра связи
EMERGENCY	0001	129(81h)-255(FFh)	1014h,1015h
PDO1(tx)	0011	385(181h)-511(1FFh)	1800h
PDO1(rx)	0100	513(201h)-639(27Fh)	1400h
PDO2(tx)	0101	641(281h)-767(2FFh)	1801h
PDO2(rx)	0110	769(301h)-895(37Fh)	1401h
PDO3(tx)	0111	897(381h)-1023(3FFh)	1802h
PDO3(rx)	1000	1025(401h)-1151(47Fh)	1402h
PDO4(tx)	1001	1153(481h)-1279(4FFh)	1803h
PDO4(rx)	1010	1281(501h)-1407(57Fh)	1403h
SDO(tx)	1011	1409(581h)-1535(5FFh)	1200h
SDO(rx)	1100	1537(601h)-1663(67Fh)	1200h
NMT Error Control	1110	1793(701h)-1919(77Fh)	1016h,1017h

Таблица 34. Обьекты для обмена точка-точка в Предопределённых Установках Связи.

Таблица 34 представляет шаблон и в практической реализации должна рассматриваться с точки зрения конкретных устройств. Предопределённые Установки Связи всегда подразумевают использование стандартного CAN фрейма с 11битовым идентификатором, даже если в сети допускаются CAN фреймы расширенного формата.
Ограничение использования COB-IDsов.
ID перечисленные в таблице 25 не должны назначаться никаким иным обьектам, и должны оставаться принадлежностью тех обьектов, которым они присвоены в Предопределённых Установках Связи.

COB-ID	Всегда принадлежит обьекту
0(000h)	Управляющие NMT обьекты
1(001h)	Зарезервирован
257(101h)-384(180h)	Зарезервирован
1409(581h)-1535(5FFh)	Предопределённый SDO(tx)
1537(601h)-1663(67Fh)	Предопределённый SDO(rx)
1760(6E0h)	Зарезервирован
1793(701h)-1919(77Fh)	Обьекты контроля ошибок NMT
2020(780h)-2047(7FFh)	Зарезервирован

Табица 35. Забронированные COB-IDs.

9.5. Обьектный словарь

9.5.1. Общая структура обьектного словаря

Данный раздел описывает подробности структуры обьектного словаря и его элементы, которые являются общими для всех устройств. Формат элемента Обьектного словаря показан в таблице 36.

Index (hex)	Object (symbolic_name)	Name	Type	Attribute	M/O

Таблица 36. Формат заголовков Обьектного Словаря.

Законченный Обьектный Словарь содержит все шесть полей показаных выше.

Столбец Index обозначает положение данного элемента в пределах OD(ObjDic). Индекс играет роль адреса при обращении к желаемому полю данных обьекта. Подиндекс здесь не описывается, но заметим всё же, что он используется для обращения к полям данных в составных обьектов, таких как массив(ARRAY), или запись(RECORD), и имеет тип UNSIGNED8.

Столбец Object является по сути классификатором обьекта в соответствии с таблицей 37, и используется для обозначения того какой такой обьект с данным отдельно взятым индексом описывается в OD.

Классификатор	Описание	Код
NULL	Элемент словаря не содержит полей данных.	0
DOMAIN	Огромная переменная набитая данными, напимер исполняемый программный код.	2
DEFTYPE	Обозначает что в данном элементе определён тип данных такой как BOOLEAN, UNSIGNED16, REAL32, и тп.	5
DEFSTRUCT	Обозначает что определяется новый тип записи(составной тип), напимер параметр PDO-отображение(21h).	6
VAR	Отдельная переменная, такая как UNSIGNED8, BOOLEAN, REAL64, INTEGER16, и тп.	7
ARRAY	Обьект с множественными полями данных, где каждый элемент переменная простого типа, того же самого что и остальные переменные, другими словами массив. Элемент с подиндексом 0 имеет тип UNSIGNED8 и не является частью массива.	8
RECORD	Запись. Обьект с множественными полями данных, где каждый элемент может иметь индивидуальный тип(обязательно простой тип). Элемент с подиндексом 0 имеет тип UNSIGNED8 и не является частью данных этого обьекта.	9

Таблица 37. Классификаторы обьектов обьектного словаря.

Столбец Name содержит простое текстовое описание функций данного обьекта.

Столбец Type даёт информацию о типе обьекта. Может быть одним из предопределённых типов: BOOLEAN, REALx, UNSIGNEDx, visible string, time-of-day, time-diference и DOMAIN(см. гл. 9.1). Помимо этого может быть предопределённый составной тип PDO-отображение, и так же другие предопределённые типы определённы производителем или профилем устройства. Не допустимы такие составные типы как записи записей или массивы записей. Если указанный тип является

массивом или записью, то подиндекс используется для указания компоненты такого составного обьекта.

Столбец Attribute определяет права доступа с точки зрения из шины в устройство(таблица 38).

Аттрибут	Описание
rw	разрешены чтение и запись
wo	только запись
ro	только чтение
const	только чтение, значение никогда не меняется

Таблица 38. Атрибут доступа к обьекту данных.

Допускается чтобы обьекты указанные в описании Обьектного словаря как rw, были реализованы как ro, исключения определяются в подробном описании обьектов.

Столбец M/O определяет является ли наличие обьекта в составе Обьектного словаря обязательным(M) или нет(O). Однако поддержка некоторых обьектов требует реализации связанных с ними обьектов. Такие тесные связи оговариваются в подробном описании обьектов.

9.5.2. Составные части словаря.

Полный план Обьектного Словаря приведён в таблице 39.

Элементы с индексами 01h-1fh содержат определение стандартных типов данных. Диапазон с индексами не определён, так как зарезервирован под будущие стандартные структуры данных.

Элементы с индексами 40h-5fh оставлены для использования изготовителями – определения их комплексных типов данных.

Диапазон с индексами 60h-7fh содержит определение стандартных типов задаваемых профилем устройства, диапазон с индексами 80h-9fh содержит определение комплексных типов данных, задаваемых профилем устройства. Диапазон a0h-25fh зарезервирован для многофункциональных устройств, может содержать определения типов данных, аналогичные элементам 60h-9fh, для устройств другого профиля.

Элементы с индексами 260h-ffffh, зарезервированы для возможного будущего расширения.

Диапазон с индексами 1000h-1ffffh включает элементы с параметрами коммуникации. Данные элементы общие для всех типов устройств, независимо от профиля к которому они относятся.

Диапазон с индексами 2000h-5ffffh для определений особых профилей изготовителя.

Диапазон с индексами 6000h-9ffffh, вмещает параметры стандартных профилей устройств.

Диапазон с индексами a000h-ffffh, зарезервирован для будущих применений.

9.5.3. Описание типов данных элементов словаря.

Статические типы данных помещены в Обьектный словарь только с целью их определения. Однако, элементы с индексами могут быть отображены как для того, чтобы определить подходящее место в RPDO, так и оставить место как неиспользуемое в данном устройстве. Индексы 0009h-000bh, 000fh не могут быть отображены в PDO.

Index	Object	Name
0001	DEFTYPE	BOOLEAN
0002	DEFTYPE	INTEGER8
0003	DEFTYPE	INTEGER16
0004	DEFTYPE	INTEGER32
0005	DEFTYPE	UNSIGNED8
0006	DEFTYPE	UNSIGNED16
0007	DEFTYPE	UNSIGNED32
0008	DEFTYPE	REAL32
0009	DEFTYPE	VISIBLE_STRING
000a	DEFTYPE	OCTET_STRING
000b	DEFTYPE	UNICODE_STRING
000c	DEFTYPE	TIME_OF_DAY
000d	DEFTYPE	TIME_DIFFERENCE
000e	RESERVED	
000f	DEFTYPE	DOMAIN
0010	DEFTYPE	INTEGER24
0011	DEFTYPE	REAL64
0012	DEFTYPE	INTEGER40
0013	DEFTYPE	INTEGER48
0014	DEFTYPE	INTEGER56
0015	DEFTYPE	INTEGER64
0016	DEFTYPE	UNSIGNED24
0017	RESERVED	
0018	DEFTYPE	UNSIGNED40
0019	DEFTYPE	UNSIGNED48
001a	DEFTYPE	UNSIGNED56
001b	DEFTYPE	UNSIGNED64
001c-001f	RESERVED	
0020	DEFSTRUCT	PDO_COMM_PAR
0021	DEFSTRUCT	PDO_MAPPING
0022	DEFSTRUCT	SDO_PARAMETER
0023	DEFSTRUCT	IDENTITY
0024-003f	RESERVED	
0040-005f	DEFSTRUCT	Комплексные типы данных, определённые изготовителем.
0060-007f	DEFTYPE	Стандартные типы данных определяемые профилем устройства 0.
0080-009f	DEFSTRUCT	Составные типы данных определяемые профилем устройства 0.
00a0-00bf	DEFTYPE	Стандартные типы данных определяемые профилем устройства 1.
00c0-00df	DEFSTRUCT	Составные типы данных определяемые профилем устройства 1.
00e0-00ff	DEFTYPE	Стандартные типы данных определяемые профилем устройства 2.
0100-011f	DEFSTRUCT	Составные типы данных определяемые профилем устройства 2.
0120-013f	DEFTYPE	Стандартные типы данных определяемые профилем устройства 3.
0140-015f	DEFSTRUCT	Составные типы данных определяемые профилем устройства 3.
0160-017f	DEFTYPE	Стандартные типы данных определяемые профилем устройства 4.
0180-019f	DEFSTRUCT	Составные типы данных определяемые профилем устройства 4.
01a0-01bf	DEFTYPE	Стандартные типы данных определяемые профилем устройства 5.
01c0-01df	DEFSTRUCT	Составные типы данных определяемые профилем устройства 5.
01e0-01ff	DEFTYPE	Стандартные типы данных определяемые профилем устройства 6.
0200-021f	DEFSTRUCT	Составные типы данных определяемые профилем устройства 6.
0220-023f	DEFTYPE	Стандартные типы данных определяемые профилем устройства 7.
0240-025f	DEFSTRUCT	Составные типы данных определяемые профилем устройства 7.

Таблица 39. Определение типов данных в Обьектном словаре

Подробности представления типов данных детально описаны в гл.9.1. Каждому устройству не обязательно поддерживать все определённые типы. Устройство должно поддерживать только те типы данных, которые используются в обьектах с индексами 1000h-9fffh. Предопределённые комплексные типы данных размещаются после стандартных типов. На данный момент определены 4 таких (т.е. составных) типа: запись Параметра связи PDO(PDO_COMMUNICATION_PARAMETER), запись PDO отображения(PDO_MAPPING), запись SDO параметра(SDO_PARAMETER), запись идентификатора(IDENTITY). Эти типы помещены в словарь под индексами 20h, 21h, 22h, 23h.

Для устройств или профилей устройств которые являются многофункциональными, т.е. как несколько независимых контроллеров, расширен механизм определения статических и составных типов, и для каждого виртуального устройства(всего до 8ми устройств) отведён диапазон в 40h элементов(040h-25fh).

Устройство может, дополнительно, предоставлять размер стандартных типов данных по чтению индексов элементов их определяющих, размер представляется как величина типа UNSIGNED32. Например, по индексу 000ch(TIME_OF_DAY) содержится значение 30h=48дес, так как данное типа TIME_OF_DAY представляется как битовая последовательность длиной 48 бит. Если длина обьекта переменная, как например, 000fh-DOMAIN, то элемент содержит 0.

Для поддерживаемых составных типов данных устройство дополнительно может сообщать их структуру, путём чтения индексов соответствующих этим типам элементов. Элемент с подиндексом 0 содержит количество полей в этом типе, подиндексы с 1 по 254 представляются как UNSIGNED8 и адресуют типы в соответствии с таблицей 39. Например элемент с индексом 20h, описывающий структуру параметра связи PDO, выглядит как(см так же обьекты с индексами 1400h-15ffh):

Подиндекс	**Значение**	**Описние**
0h	04h	4 подэлемента, описываемых следующими подиндексами
1h	07h	UNSIGNED32
2h	05h	UNSIGNED8
3h	06h	UNSIGNED16
4h	05h	UNSIGNED8

Таблица 40. Пример описания комплексного типа данных.

9.5.3.1. Состав структурных элементов Обьектного Словаря.

Если элемент(индекс) Словаря содержит несколько подэлементов(подиндексов), то подиндекс 0 содержит количество следующих за ним подиндексов, остальные(за исключением подиндекса 255) описывают подэлементы и представляются как UNSIGNED8. Подиндекс ffh, описывает тип данных и тип элемента в обьекте. Он представляется как UNSIGNED32, и имеет структуру описанную ниже.

Биты	31-16	15-8	7-0
Значение	Зарезервировано всегда 0	Тип данных (таблица 39)	Обьект (таблица 37)
Представление	-	UNSIGNED8	UNSIGNED8

Рисунок 51:). Структура подиндекса ffh.

Поддержка подиндекса 0xff необязательна. Если такая поддержка имеет место без исключений, и известна структура составных типов, то это позволяет скачать полную структуру обьектного словаря.

9.5.4. Описение предопределённых составных типов данных.

Данный пункт описывает структуру предопределённых составных типов используемых для связи. Диапазон значений и смысл величин подробно описан в описании обьектов имеющих данные типы.

9.5.4.1. Описатель записи типа Параметр связи PDO.

Индекс	Подиндекс	Название поля	Тип данных
0020h	0h	Количество подэлементов в записи	UNSIGNED8
	1h	COB-ID	UNSIGNED32
	2h	тип передачи	UNSIGNED8
	3h	время подавления повторной передачи	UNSIGNED16
	4h	зарезервирован	UNSIGNED8
	5h	время ожидания события	UNSIGNED16

Таблица 41.

9.5.4.2. Описатель записи типа Параметр отображения PDO.

Индекс	Подиндекс	Название поля	Тип данных
0021h	0h	Количество отображаемых элементов	UNSIGNED8
	1h	1й отображённый обьект	UNSIGNED32
	2h	2й отображённый обьект	UNSIGNED32
...	...	...	...
	40h	64й отображённый обьект	UNSIGNED32

Таблица 42.

9.5.4.3. Описатель записи типа Параметр SDO.

Индекс	Подиндекс	Название поля	Тип данных
0022h	0h	Количество подэлементов в записи	UNSIGNED8
	1h	COB-ID клиент->сервер	UNSIGNED32
	2h	COB-ID сервер->клиент	UNSIGNED32
	3h	ID узла клиента/сервера	UNSIGNED8

Таблица 43.

9.5.4.4. Описатель записи типа Идентификатор.

Индекс	Подиндекс	Название поля	Тип данных
0023h	0h	Количество подэлементов в записи	UNSIGNED8
	1h	ID-Изготовителя(Vendor ID)	UNSIGNED32
	2h	Код изделия	UNSIGNED8
	3h	Номер издания	UNSIGNED16
	4h	Серийный номер	UNSIGNED8

Таблица 44.

9.6. Описание профиля связи.

9.6.1. Детальное описание обьектов

Обьекты всех профилей устройств, интерфейсов и приложений основанных на данном профиле связи, должны описываться по приведённому шаблону:

ОПИСЕНИЕ ОБЬЕКТА

Index	Индекс
Name	Символьное имя обьекта
Object code	Классификатор обьекта
Data type	Тип данных обьекта
Category	Обязателен или нет

Таблица 45. Формат описания обьекта.

Object code должен содержать одно из значений приведённых в таблице 37. Для лучшей читаемости, описание обьекта содержит поле Name – символьное имя обьекта, в OD естественно, такого поля нет.

ОПИСАНИЕ ЭЛЕМЕНТА.

Sub-index	Количество описанных подэлементов(только для полей структур, массивов и записей)
Description	Описание поля (только для полей структур, массивов и записей)
Data type	Тип данных поля (только для полей структур, массивов и записей)
Entry category	Поле в составе обьекта – обязательно или нет
Access	ro, wo, rw, const(см табл 38.) способы доступа к обьекту.
PDO mapping	Optional/Default/No – может ли данный обьект отображён в PDO. Optional: обьект может быть отображён в PDO. Default: обьект по умолчанию является частью PDO отображения (см. профиль устройства). No: обьект не может быть отображён в PDO.
Value range	Диапазон возможных значений. Если может быть любым для данного типа переменной, то указывается имя типа.
Defaul value	No: значение по умолчанию не оговаривается в данной спецификации. значение: Значение загружаемое в устройство во время его инициализации.

Таблица 46. Формат описания значений обьекта.

Для простых переменных описание значений приводится без указания Sub-index, Description, Data type и Entry category. Для составных элементов описание значений должно быть приведено для каждого поля(подэлемента).

9.6.2. Обзор элементов описания связи Обьектного Словаря.

В таблице 47 приведён краткий обзор элементов Обьектного Словаря, определённых профилем связи. Поле Access приведённое в таблице может варьироваться для отдельныз подэлементов составных обьектов, подробнее описано в описании этих обьектов.

Index (hex)	Object (symbolic name)	Name	Type	Attr.	M/O
1000	VAR	Тип устройства	UNSIGNED32	ro	**M**
1001	VAR	Регистр ошибок	UNSIGNED8	ro	**M**
1002	VAR	Регистр статуса от производителя	UNSIGNED32	ro	**O**
1003	ARRAY	Предопределённое поле регистрации ошибок	UNSIGNED32	ro	**O**
1004	Зарезервировано из соображений совместимости				
1005	VAR	Идентификатор SYNC обьекта	UNSIGNED32	rw	**O**
1006	VAR	Период цикла связи	UNSIGNED32	rw	**O**
1007	VAR	Длительность окна синхронных PDO	UNSIGNED32	rw	**O**
1008	VAR	Название изготовителя устройства	VIS.STRING	const	**O**
1009	VAR	Версия аппаратной части	VIS.STRING	const	**O**
100a	VAR	Версия программного обеспечения	VIS.STRING	const	**O**
100b	Зарезервировано из соображений совместимости				
100c	VAR	Период повторения сторожевых запросов	UNSIGNED16	rw	**O**
100d	VAR	Коэффициент пропорциональности сторожевого ожидания	UNSIGNED8	rw	**O**
100e	Зарезервировано из соображений совместимости				
100f	Зарезервировано из соображений совместимости				
1010	ARRAY	Управление сохранением параметров	UNSIGNED32	rw	**O**
1011	ARRAY	Восстановление значения параметров по умолчанию	UNSIGNED32	rw	**O**
1012	VAR	Идентификатор TIME обьекта	UNSIGNED32	rw	**O**
1013	VAR	Высокоточная временная отметка	UNSIGNED32	rw	**O**
1014	VAR	Идентификатор EMCY обьекта	UNSIGNED32	rw	**O**
1015	VAR	Время подавления следующего EMCY	UNSIGNED16	rw	**O**
1016	ARRAY	Время ожидания контрольного тактирования	UNSIGNED32	rw	**O**
1017	VAR	Период контрольного тактирования	UNSIGNED16	rw	**O**
1018	RECORD	Обьект идентификации изделия	Identity (23h)	ro	**M**
1019		Зарезервировано			
:::::	:::::	:::::	:::::	:::	:::
11ff		Зарезервировано			
Параметры серверных SDO					
1200	RECORD	Параметр серверного SDO №1	SDOPar(22h)	ro	**O**
1201	RECORD	Параметр серверного SDO №2	SDOPar(22h)	rw	**M/O****
:::::	:::::	:::::	:::::	:::	:::
127f	RECORD	Параметр серверного SDO №128	SDOPar(22h)	rw	**M/O****
Параметры клиентских SDO					
1280	RECORD	Параметр клиентского SDO №1	SDOPar(22h)	ro	**M/O****
1281	RECORD	Параметр клиентского SDO №2	SDOPar(22h)	rw	**M/O****
:::::	:::::	:::::	:::::	:::	:::
12ff	RECORD	Параметр клиентского SDO №128	SDOPar(22h)	rw	**M/O****
1300		Зарезервировано			
:::::	:::::	:::::	:::::	:::	:::

13ff		Зарезервировано			
Параметры связи RPDO					
1400	RECORD	Параметр связи RPDO №1	PDO Com.Par (20h)	rw	**M/O***
1401	RECORD	Параметр связи RPDO №2	PDO Com.Par (20h)	rw	**M/O***
:::::	:::::	:::::	:::::	:::	:::
15ff	RECORD	Параметр связи RPDO №512	PDO Com.Par (20h)	rw	**M/O***
Параметры отображения RPDO					
1600	RECORD	Параметр отображения RPDO №1	PDO Mapping (21h)	rw	**M/O***
1601	RECORD	Параметр отображения RPDO №2	PDO Mapping (21h)	rw	**M/O***
:::::	:::::	:::::	:::::	:::	:::
17ff	RECORD	Параметр отображения RPDO №512	PDO Mapping (21h)	rw	**M/O***
Параметры связи TPDO					
1800	RECORD	Параметр связи TPDO №1	PDO Com.Par (20h)	rw	**M/O***
1801	RECORD	Параметр связи TPDO №2	PDO Com.Par (20h)	rw	**M/O***
:::::	:::::	:::::	:::::	:::	:::
19ff	RECORD	Параметр связи TPDO №512	PDO Com.Par (20h)	rw	**M/O***
Параметры отображения TPDO					
1a00	RECORD	Параметр отображения TPDO №1	PDO Mapping (21h)	rw	**M/O***
1a01	RECORD	Параметр отображения TPDO №2	PDO Mapping (21h)	rw	**M/O***
:::::	:::::	:::::	:::::	:::	:::
1bff	RECORD	Параметр отображения TPDO №512	PDO Mapping (21h)	rw	**M/O***

* Если устройство поддерживает PDOsы, то соответствующие параметры связи PDO и отображения PDO обязательны, они могут быть с атрибутом только для чтения
** Если устройство поддерживает SDOsы, соответствующие параметры SDO обязательны

9.6.3. Подробное описание элементов профиля связи Обьектного Словаря.

Обьект 1000h: Тип устройства.

Содержит информацию о типе устройства. Обьект с индексом 1000h описывает тип устройства и его функциональность. Он составляется из 16-битного поля, которое описывает использующийся профиль устройства, и второе 16-битное поле, которое даёт дополнительную информацию о расширенных возможностях прибора. Параметр дополнительной информации определяется профилем устройства, и его описание не входит в рамки данной спецификации, это определено в соответствующем профиле устройств. Значение 0000h сообщает что данное устройство не соответствует никакому стандартному профилю. Для многофункциональных модулей параметр дополнительной информации содержит ffffh, а профиль устройства указанный в указателе профиля обьекта 1000h есть профиль первого устройства обьектного словаря. Все остальные устройства многофункционального модуля идентифицируют свои профили с помощью элементов 67ffh+x*800h, где x это внутренний номер этого устройства в многофункциональном модуле(0-7). Этот элемент описывает тип устройства описанного предшествующим элементом конфигурации.
ОПИСЕНИЕ ОБЬЕКТА

Index	1000h
Name	Тип устройства
Object code	VAR
Data type	UNSIGNED32
Category	Обязателен(M)

ОПИСАНИЕ ЭЛЕМЕНТА.

Access	ro
PDO mapping	No
Value range	UNSIGNED32
Defaul value	No

MSB LSB

Дополнительная информация	Номер профиля устройства

Рисунок 52. Структура обьекта 1000h – Параметр типа устройства.

Обьект 1001h: Регистр ошибок.

Устройство может отображать флагами в этом обьекте внутренние ошибки. Этот элемент обязателен для всех устройств – он часть аврального обьекта.

ОПИСЕНИЕ ОБЬЕКТА

Index	1001h
Name	Регистр ошибки
Object code	VAR
Data type	UNSIGNED8
Category	Обязателен(M)

ОПИСАНИЕ ЭЛЕМЕНТА.

Access	ro
PDO mapping	Optional
Value range	UNSIGNED8
Defaul value	No

Бит	M/O	Значение
0	M	Общая ошибка
1	O	Ток
2	O	Напряжение
3	O	Температура
4	O	Ошибка связи(перегрузка, некорректное состояние)
5	O	Определяется профилем устройства
6	O	Зарезервирован(всегда 0)
7	O	Определяется изготовителем

Таблица 48. Структура обьекта 1001h – Регистр ошибки.

Если какой либо бит установлен, то это указывает на то что произошла соответствующая ошибка. Есть одна ошибка, которая должна указываться в обязательном порядке – это Общая ошибка, остальные если требуется профилем устройства. Общая ошибка устанавливается в случае любой из указанных ошибкок.

Обьект 1002h: Регистр статуса от изготовителя.

ОПИСЕНИЕ ОБЬЕКТА

Index	1002h
Name	Регистр статуса от изготовителя
Object code	VAR
Data type	UNSIGNED32
Category	Необязателен(O)

ОПИСАНИЕ ЭЛЕМЕНТА.

Access	ro
PDO mapping	Optional
Value range	UNSIGNED32
Defaul value	No

Обьект 1003h: Предопределённое поле регистрации ошибок.

Данный обьект хранит информацию об ошибках произошедших в устройстве и сигнализирует о них при помощи аврального обьекта. Так же он предоставляет хронологию ошибок.

1. Элемент с подиндексом 0 содержит количество актуальных на данный момент ошибок хранящихся в виде массива, начиная с подиндекса 1.

2. Каждая новая ошибка запоминается с подиндексом 1, все более старые ошибки сдвигаются вниз по списку.

3. Запись нуля в элемент с подиндексом 0 удаляет всю историю ошибок(очищает массив). Запись в данный элемент значений отличных от нуля не допускается, это приведёт к ошибке с кодом 0609 0030h.

4. Код ошибки представляются как величина UNSIGNED32, и содержит 16ти битный код ошибки(младшие 2 байта) и 16ти битное поле дополнительной информации об ошибках, которая определяется изготовителем(старшие 2 байта).

MSB LSB

Дополнительная информация	Код ошибки

Рисунок 53. Структура предопределённого поля ошибок.

ОПИСЕНИЕ ОБЬЕКТА

Index	1003h
Name	Предопределённое поле ошибок
Object code	ARRAY
Data type	UNSIGNED32
Category	Необязателен(O)

ОПИСАНИЕ ЭЛЕМЕНТА.

Sub-index	0h
Description	Количество ошибок
Entry category	Обязателен(M)
Access	rw
PDO mapping	No
Value range	0-254
Defaul value	0
Sub-index	1h

Description	Стандартное поле регистации ошибок
Entry category	Необязателен(O)
Access	ro
PDO mapping	No
Value range	UNSIGNED32
Defaul value	No

Sub-index	2h-FEh
Description	Стандартное поле регистрации ошибок
Entry category	Необязателен(O)
Access	ro
PDO mapping	No
Value range	UNSIGNED32
Defaul value	No

Обьект 1005h. Идентификатор SYNC обьекта.

Помимо того что этот обьект хранит идентификатор SYNC обьекта, он определяет является ли данное устройство генератором синхрососообщений.

UNSIGNED32

MSB LSB

Биты	31	30	29	28-11	10-0
11битный формат	X	0/1	0	00 0000 0000 0000 0000	11-битный идентификатор
29битный формат	X	0/1	1	29-битный идентификатор	

Рисунок 54. Структура Идентификатора COB SYNC.

Номер бита	Значение	Описание
31(MSB)	X	Не определён
30	0 1	Устройство не генерирует синхрососообщения. Устройство генерирует синхрососообщения
29	0 1	11-битный идентификатор(CAN 2.0A) 29-битный идентификатор(CAN 2.0B)
28-11	0 X	Если бит 29 = 0 Если бит 29 ≠ 0, содержит биты 28-11 29 битного идентификатора
10-0(LSB)	X	биты 10-0 идентификатора

Таблица 49. Описание элемента SYNC COB-ID.

Биты 29, 30 могут быть статическими(не изменяемыми). Если устройство в принципе не способно генерировать синхроимпульсы, то на попытку установить бит 30, оно должно ответить сообщением об ошибке с кодом 0609 0030h. Устройства поддерживающие толко стандартный CAN фрейм, должны либо игнорировать попытки переписать бит 29, либо возвращать код ошибки 0609 0030h. Передача первого синхрообьекта начинается через однин период синхронизации, после установки бита 30 в 1. Недопустимо менять биты 29-0 если установлен бит 30.

ОПИСЕНИЕ ОБЬЕКТА

Index	1005h
Name	Идентификатор SYNC обьекта SYNC COB-ID
Object code	VAR
Data type	UNSIGNED32
Category	Условие Обязателен(M), если поддерживается обмен синхронными PDO.

ОПИСАНИЕ ЭЛЕМЕНТА.

Access	rw
PDO mapping	No
Value range	UNSIGNED32
Defaul value	80h или 8000 0080h

Обьект 1006h: Период цикла связи.

Данный обьект задаёт период цикла связи (повторения сообщений SYNC) в микросекундах. Если значение этого параметра 0 если синхронизация не осуществляется. Если для синхрогенератора период цикла связи меняется в состояние отличное от нуля, то передача нового синхросообщения произойдёт спустя один цикл с новым периодом.

ОПИСЕНИЕ ОБЬЕКТА

Index	1006h
Name	Период цикла связи
Object code	VAR
Data type	UNSIGNED32
Category	Условие Обязателен(M), для синхрогенератора.

ОПИСАНИЕ ЭЛЕМЕНТА.

Access	rw
PDO mapping	No
Value range	UNSIGNED32
Defaul value	0

Обьект 1007h: Длительность окна синхронных PDO.

Содержит длительность временного окна для синхронных PDO в микросекундах. Значение о, если не используется обмен синхронными PDO.

ОПИСЕНИЕ ОБЬЕКТА

Index	1007h
Name	Длительность окна синхронных PDO
Object code	VAR
Data type	UNSIGNED32
Category	Необязателен

ОПИСАНИЕ ЭЛЕМЕНТА.

Access	rw
PDO mapping	No
Value range	UNSIGNED32
Defaul value	0

Обьект 1008h: Название изготовителя устройства.

ОПИСЕНИЕ ОБЬЕКТА

Index	1008h
Name	Название изготовителя устройства
Object code	VAR
Data type	VISIBLE_STRING
Category	Необязателен

ОПИСАНИЕ ЭЛЕМЕНТА.

Access	const
PDO mapping	No
Value range	No
Defaul value	No

Обьект 1009h: Версия аппаратной части.

ОПИСЕНИЕ ОБЬЕКТА

Index	1009h
Name	Версия аппаратной части
Object code	VAR
Data type	VISIBLE_STRING
Category	Необязателен

ОПИСАНИЕ ЭЛЕМЕНТА.

Access	const
PDO mapping	No
Value range	No
Defaul value	No

Обьект 100ah: Версия программного обеспечения.

ОПИСЕНИЕ ОБЬЕКТА

Index	100ah
Name	Версия программного обеспечения
Object code	VAR
Data type	VISIBLE_STRING
Category	Необязателен

ОПИСАНИЕ ЭЛЕМЕНТА.

Access	const
PDO mapping	No
Value range	No
Defaul value	No

Обьект 100ch: Период повторения сторожевых запросов.

Обьекты 100ch и 100dh содержат параметры для механизма караула узлов: период повторения сторожевых запросов(ПСО, в миллисекундах) и коэффициент пропорциональности сторожевого ожидания(КПСО). ПСО*КПСО = Время сторожевого ожидания, если равно 0, то данный механизм не используется. (см п 9.2.6.)

ОПИСЕНИЕ ОБЬЕКТА

Index	100ch
Name	Период повторения сторожевых запросов
Object code	VAR
Data type	UNSIGNED16
Category	Условие Обязателен(M), если не поддерживается механизм контрольного тактирования

ОПИСАНИЕ ЭЛЕМЕНТА.

Access	rw ro, если не поддерживается механизм караула узлов
PDO mapping	No
Value range	UNSIGNED16
Defaul value	0

Обьект 100dh: Коэффициент пропорциональности сторожевого ожидания.

ОПИСЕНИЕ ОБЬЕКТА

Index	100dh
Name	Коэффициент пропорциональности сторожевого ожидания
Object code	VAR
Data type	UNSIGNED8
Category	Условие Обязателен(M), если не поддерживается механизм контрольного тактирования

ОПИСАНИЕ ЭЛЕМЕНТА.

Access	rw ro, если не поддерживается механизм караула узлов
PDO mapping	No
Value range	UNSIGNED8
Defaul value	0

Обьект 1010h: Управление сохранением параметров.

Данный обьект поддерживает сохранение параметров в энергонезависимой памяти. Путём чтения данного элемента можно получить информацию о способностях сохранения параметров данным устройством. Может быть несколько групп параметров:
Подиндекс 0 содержит максимальный подиндекс который поддерживается.
Подиндекс 1 ссылается на все параметры устройства, которые могут быть сохранены.
Подиндекс 2 ссылается на параметры связи, которые могут быть сохранены (определяемые изготовителем параметры из диапазона 1000-1fffh)
Подиндекс 3 ссылается на параметры приложения, которые могут быть сохранены (определяемые изготовителем параметры из диапазона 6000-9fffh)
В подиндексах 4-127 изготовитель может задать отдельные параметры по своему усмотрению.
Подиндексы 128-254 зарезервированы для будущего использования.

Во избежании сохранения параметров по ошибке, сохранение будет начато только если особое ключевое слово будет записано в соответствующий подиндекс. Данное слово – «save».

Signature	MSB			LSB
ISO 8859 ("ASCII")	e	v	a	s
hex	65h	76h	61h	73h

Рисунок 55. Ключевое слово для сохранения параметров.

По приёму корректного ключевого слова в соответствующий подиндекс, устройство производит сохранение параметров и после подтверждает SDO передачу(подтверждение инициации SDO загрузки). Если запоминание не удалось, то устройство должно вместо подтверждения вернуть код Аварийного завершения SDO трансферта (код – 0606 0000h).

Если записано неверное ключевое слово, то устройство отказывает в запоминании и отвечает аварийным завершением SDO трансферта(код 0800 002xh).

При чтении из соответствующего элемента устройство предоставляет о своих возможностях запоминания в формате:

	UNSIGNED32		
	MSB		LSB
bits	*31-2*	*1*	*0*
	reserved (=0)	0/1	0/1

Рисунок 56. Структура описания возможностей сохранения в энергонезависимой памяти.

Номер бита	Значение	Описание
31-2	0	Зарезервировано(всегда 0)
1	0 1	Устройство не сохраняет параметры автоматически Устройство сохраняет параметры автоматически
0	0 1	Устройство не сохраняет параметры по команде Устройство сохраняет параметры по команде

Таблица 50. Пояснения к рисунку 56.

ОПИСЕНИЕ ОБЬЕКТА

Index	1010h
Name	Управление сохранением параметров
Object code	ARRAY
Data type	UNSIGNED32
Category	Необязателен(O)

ОПИСАНИЕ ЭЛЕМЕНТА.

Sub-index	0h
Description	Количество сохраняемых групп
Entry category	Обязателен(M)
Access	ro
PDO mapping	No
Value range	1h – 7fh
Defaul value	No

Sub-index	1h
Description	Сохранить все параметры.
Entry category	Обязателен(M)
Access	rw
PDO mapping	No
Value range	UNSIGNED32 Рис55 при записи Рис 56 при чтении
Defaul value	No

Sub-index	2h
Description	Сохранить настройки связи
Entry category	Необязателен(O)
Access	rц
PDO mapping	No
Value range	UNSIGNED32 Рис55 при записи Рис 56 при чтении
Defaul value	No

Sub-index	3h
Description	Сохранить параметры приложения
Entry category	Необязателен(O)
Access	rw
PDO mapping	No
Value range	UNSIGNED32 Рис55 при записи Рис 56 при чтении
Defaul value	No

Sub-index	4h
Description	Сохранение групп и отдельных параметров определённых изготовителем.
Entry category	Необязателен(O)
Access	rw
PDO mapping	No
Value range	UNSIGNED32 Рис55 при записи Рис 56 при чтении
Defaul value	No

Обьект 1011h: Восстановление значения параметров по умолчанию.

При помощи этого обьекта восстанавливаются значения по умолчанию, указанные в профиле устройства или связи. Путём чтения соответствующих элементов обьекта можно узнать о способности устройства восстанавливать значения. Различают несколько групп параметров.
Подиндекс 0 содержит максимальный подиндекс который поддерживается.
Подиндекс 1 ссылается на все параметры устройства, которые могут быть восстановлены.
Подиндекс 2 ссылается на параметры связи, которые могут быть восстановлены (определяемые изготовителем параметры из диапазона 1000-1fffh)
Подиндекс 3 ссылается на параметры приложения, которые могут быть восстановлены (определяемые изготовителем параметры из диапазона 6000-9fffh)
В подиндексах 4-127 изготовитель может задать отдельные параметры по своему усмотрению.
Подиндексы 128-254 зарезервированы для будущего использования.

Во избежании восстановления параметров по ошибке, оно будет выполнено только если особое ключевое слово будет записано в соответствующий подиндекс. Данное слово – «save».

Signature	MSB			LSB
ASCII	d	a	o	l
hex	64h	61h	6Fh	6Ch

Рисунок 57. Ключевое слово для восстановления параметров.

По приёму корректного ключевого слова в соответствующий подиндекс, устройство производит восстановление параметров и после подтверждает SDO передачу(подтверждение инициации SDO загрузки). Если восстановление не удалось, то устройство должно вместо подтверждения вернуть код Аварийного завершения SDO трансферта (код – 0606 0000h).

Если записано неверное ключевое слово, то устройство отказывает в запоминании и отвечает аварийным завершением SDO трансферта(код 0800 002xh).

Восстановленные параметры примут значения по умолчанию после сброса(сброса связи для параметров указанных подиндексом 2h и сброс узла для всех остальных) или следующего включения.

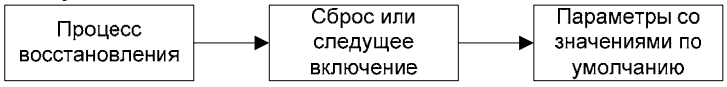


Рисунок 58. Процедура восстановления.

При чтении из соответствующего элемента устройство предоставляет о своих возможностях восстановления параметров по умолчанию в формате:

UNSIGNED32

	MSB	LSB
bits	*31-1*	*0*
	reserved (=0)	0/1

Рисунок 59. Структура описания возможностей восстановления значений по умолчанию.

Номер бита	Значение	Описание
31-1	0	Зарезервировано(всегда 0)
0	0 1	Устройство не восстановливает значения по умолчанию Устройство восстановливает значения по умолчанию

Таблица 51. Пояснения к рисунку 59.

ОПИСЕНИЕ ОБЬЕКТА

Index	1011h
Name	Восстановление значения параметров по умолчанию
Object code	ARRAY
Data type	UNSIGNED32
Category	Необязателен(O)

ОПИСАНИЕ ЭЛЕМЕНТА.

Sub-index	0h
Description	Количество восстанавливаемых групп
Entry category	Обязателен(M)
Access	ro
PDO mapping	No
Value range	1h – 7fh
Defaul value	No

Sub-index	1h
Description	Восстановить все параметры.
Entry category	Обязателен(M)
Access	rw
PDO mapping	No
Value range	UNSIGNED32 Рис 57 при записи Рис 59 при чтении
Defaul value	No

Sub-index	2h
Description	Восстановить настройки связи
Entry category	Необязателен(O)
Access	rw
PDO mapping	No
Value range	UNSIGNED32 Рис57 при записи Рис 59 при чтении
Defaul value	No

Sub-index	3h
Description	Восстановить параметры приложения
Entry category	Необязателен(O)
Access	rw
PDO mapping	No
Value range	UNSIGNED32 Рис57 при записи Рис 59 при чтении
Defaul value	No

Sub-index	4h
Description	Восстановление групп и отдельных параметров определённых изготовителем.
Entry category	Необязателен(O)
Access	rw
PDO mapping	No
Value range	UNSIGNED32 Рис57 при записи Рис 59 при чтении
Defaul value	No

Обьект 1012h: Идентификатор TIME обьекта.

Помимо того что этот обьект хранит идентификатор TIME_STAMP обьекта, он определяет является ли данное устройство генератором таких обьектов.

	UNSIGNED32				
	MSB				LSB
bits	*31*	*30*	*29*	*28-11*	*10-0*
11-bit-ID	0/1	0/1	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	11-bit Identifier
29-bit-ID	0/1	0/1	1	29 -bit Identifier	

Рисунок 60. Структура Структура Идентификатора COB TIME.

Номер бита	Значение	Описание
31(MSB)	0 1	Устройство не потребляет TIME сообщения Устройство потребляет TIME сообщения
30	0 1	Устройство не генерирует TIME сообщения. Устройство генерирует TIME сообщения

29	0 1	11-битный идентификатор(CAN 2.0A) 29-битный идентификатор(CAN 2.0B)
28-11	0 X	Если бит 29 = 0 Если бит 29 ≠ 0, содержит биты 28-11 29 битного идентификатора
10-0(LSB)	X	биты 10-0 идентификатора

Таблица 49. Описание элемента TIME COB-ID.

Биты 29, 30 могут быть статическими(не изменяемыми). Если устройство в принципе не способно генерировать TIME сообщения, то на попытку установить бит 30, оно должно ответить сообщением об ошибке с кодом 0609 0030h. Устройства поддерживающие толко стандартный CAN фрейм, должны либо игнорировать попытки переписать бит 29, либо возвращать код ошибки 0609 0030h. Недопустимо менять биты 29-0 если установлен бит 30.

ОПИСЕНИЕ ОБЬЕКТА

Index	1012h
Name	Идентификатор TIME обьекта TIME COB-ID
Object code	VAR
Data type	UNSIGNED32
Category	Необязателен(O)

ОПИСАНИЕ ЭЛЕМЕНТА.

Access	rw
PDO mapping	No
Value range	UNSIGNED32
Defaul value	100h

Обьект 1013h: Высокоточная временная отметка.

Обьект содержит временную отметку с разрешением 1 мкС(см п.9.3.2.). Он может быть отображён в PDO, чтобы определить сообщение высокоточной временной отметки. Заметим, что сейчас тип данных фиксирован (TIME_STAMP), в дальнейшем может приветствоваться использование типа подходящего приложению.

ОПИСЕНИЕ ОБЬЕКТА

Index	1013h
Name	Высокоточная временная отметка
Object code	VAR
Data type	UNSIGNED32
Category	Необязателен(O)

ОПИСАНИЕ ЭЛЕМЕНТА.

Access	rw
PDO mapping	Может быть
Value range	UNSIGNED32
Defaul value	0

Обьект 1014h: Идентификатор аврального обьекта – срочное сообщение EMCY COB-ID.

	UNSIGNED32				
	MSB				LSB
bits	*31*	*30*	*29*	*28-11*	*10-0*
11-bit-ID	0/1	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	11-bit Identifier
29-bit-ID	0/1	0	1	29 -bit Identifier	

Рисунок 61. Структура Идентификатора COB EMCY.

Номер бита	Значение	Описание
31(MSB)	0 1	EMCY создан/ действителен EMCY не создан/ недействителен
30	0	Зарезервировано, всегда 0.
29	0 1	11-битный идентификатор(CAN 2.0A) 29-битный идентификатор(CAN 2.0B)
28-11	0 X	Если бит 29 = 0 Если бит 29 ≠ 0, содержит биты 28-11 29 битного идентификатора
10-0(LSB)	X	биты 10-0 идентификатора

Таблица 49. Описание элемента EMCY COB-ID.

Устройства поддерживающие толко стандартный CAN фрейм, должны либо игнорировать попытки переписать бит 29, либо возвращать код ошибки 0609 0030h. Недопустимо менять биты 29-0 если бит 31=0.

ОПИСЕНИЕ ОБЬЕКТА

Index	1014h
Name	Идентификатор EMCY обьекта EMCY COB-ID
Object code	VAR
Data type	UNSIGNED32
Category	Условие Обязателен(M), если поддерживается механизм срочных сообщений.

ОПИСАНИЕ ЭЛЕМЕНТА.

Access	ro допускается rw
PDO mapping	No
Value range	UNSIGNED32
Defaul value	80h + ID узла

Обьект 1015h: Время подавления следующего EMCY

Через этот элемент может быть настроено время запрета передачи следующего EMCY(так как у них очень высокий приоритет, чтобы не подтормаживать остальные обьекты). Указанное значение умножается на 100 мкС.

ОПИСЕНИЕ ОБЬЕКТА

Index	1015h
Name	Время запрета передачи следующего EMCY
Object code	VAR
Data type	UNSIGNED16
Category	Необязателен

ОПИСАНИЕ ЭЛЕМЕНТА.

Access	rw
PDO mapping	No
Value range	UNSIGNED16
Defaul value	0

Обьект 1016h: Время ожидания контрольного тактирования.

Этот параметр определяет ожидаемое значение периода прихода контрольных сообщений и должен быть больше чем период контрольного тактирования в генераторе тактирования. Мониторинг начинается как только приходит первое контрольное сообщение. Если значение 0 то это означает, что контрольное тактирование не поддерживается. Значение времени задаётся в миллисекундах.

	MSB		LSB
Bits	31-24	23-16	15-0
Value	reserved (value: 00h)	Node-ID	heartbeat time
Encoded as	-	UNSIGNED8	UNSIGNED16

Рисунок 62. Структура обьекта.

При попытке задать несколько несколько контрольных элементов с одинаковым ID и временами не равными нулю, устройство вернёт код Аварийного завершения SDO трансферта 0604 0043;

ОПИСЕНИЕ ОБЬЕКТА

Index	1016h
Name	Время ожидания контрольного тактирования
Object code	ARRAY
Data type	UNSIGNED32
Category	Необязателен(O)

ОПИСАНИЕ ЭЛЕМЕНТА.

Sub-index	0h
Description	Число контролируемых источников
Entry category	Обязателен(M)
Access	ro
PDO mapping	No
Value range	1 – 127
Defaul value	No

Sub-index	1h
Description	Время ожидания от источника №1
Entry category	Обязателен(M)
Access	rw
PDO mapping	No
Value range	UNSIGNED32 Рис 62 при записи
Defaul value	0

Sub-index	2h-7f
Description	Время ожидания от источника №…
Entry category	Необязателен(O)
Access	rw
PDO mapping	No
Value range	UNSIGNED32 Рис 62 при записи
Defaul value	No

Обьект 1017h: Период контрольного тактирования.

Если значение 0 то данное устройство не генерирует никаких контрольных сообщений. Время задаётся в миллисекундах.

ОПИСЕНИЕ ОБЬЕКТА

Index	1017h
Name	Период контрольного тактирования
Object code	VAR
Data type	UNSIGNED16
Category	Условие Обязателен(M) Если требуется от устройства механизмами контрольного тактирования.

ОПИСАНИЕ ЭЛЕМЕНТА.

Access	rw
PDO mapping	No
Value range	UNSIGNED16
Defaul value	0

Обьект 1018h: Обьект идентификации изделия.

Данный обьект отражает основную информацию об этом устройстве. Код изготовителя(Vendor ID) содержит уникальный номер присваеваемый каждому изготовителю. Определяемый изготовителем Код продукта описывает разновидность прибора. Проеделяемый изготовителем Номер версии содержит версию и подверсию устройства. Версия(**major**) идентифицирует функциональность CANopen устройства, а подверсия(**minor**) - варианты реализации функциональности.

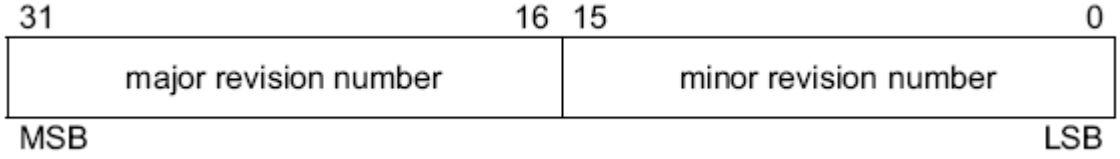


Рисунок 63. Структура номера версии.

ОПИСЕНИЕ ОБЬЕКТА

Index	1018h
Name	Обьект идентификации изделия
Object code	RECORD
Data type	Identity(23h)
Category	Обязателен(M)

ОПИСАНИЕ ЭЛЕМЕНТА.

Sub-index	0h
Description	Количество элементов
Entry category	Обязателен(M)
Access	ro
PDO mapping	No
Value range	1..4
Defaul value	No

Sub-index	1h
Description	Код изготовителя(Vendor ID)
Entry category	Обязателен(M)
Access	ro
PDO mapping	No
Value range	UNSIGNED32
Defaul value	No

Sub-index	2h
Description	Код продукта
Entry category	Необязателен(O)
Access	ro
PDO mapping	No
Value range	UNSIGNED32
Defaul value	No

Sub-index	3h
Description	Номер версии
Entry category	Необязателен(O)
Access	ro
PDO mapping	No
Value range	UNSIGNED32
Defaul value	No

Sub-index	4h
Description	Серийный номер
Entry category	Необязателен(O)
Access	ro
PDO mapping	No
Value range	UNSIGNED32
Defaul value	No

Обьект 1200h-127fh: Описатель SDO сервера.

Для описания SDOsов используемых в устройстве введён тип Параметр SDO. Этот тип данных имеет индекс 22h в Обьектном Словаре, а структура уже рассматривалась в п 9.5.4. Подиндексом 0h задаётся количество элементов в описателе SDO. Элементы 1h и 2h задают COB-ID для этого SDO. Элемент 3h указывает клиенту номер узла сервера, а серверу номер узла клиента(довольно странный параметр – номер узла определяет IDSDO, заданного в соответствии с DS301, может использоваться для вычисления элементов 1 и 2 и больше практической цели для этого параметра нет, скорее справочный характер).

UNSIGNED32

	MSB				LSB
bits	*31*	*30*	*29*	*28-11*	*10-0*
11-bit-ID	0/1	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	11-bit Identifier
29-bit-ID	0/1	0	1	29 -bit Identifier	

Рисунок 64. Структура Идентификатора COB SDO.

Номер бита	Значение	Описание
31(MSB)	0 1	SDO создан/ действителен SDO не создан/ недействителен
30	0	Зарезервировано, всегда 0.
29	0 1	11-битный идентификатор(CAN 2.0A) 29-битный идентификатор(CAN 2.0B)
28-11	0 X	Если бит 29 = 0 Если бит 29 ≠ 0, содержит биты 28-11 29 битного идентификатора
10-0(LSB)	X	биты 10-0 идентификатора

Таблица 54. Описание элемента SDO COB-ID.

SDO будет действителен только если действительны (бит 31 = = 0) оба идентификатора – клиент->сервер и сервер->клиент. Устройства, поддерживающие только стандартный CAN фрейм, при попытке установить бит 29, должны возвращать код ошибки 0609 0030h. Если устройство обрабатывает более чем один SDO сервер, то default SDO должен располагаться в словаре под индексом 1200h как первый SDO сервер. Этот элемент имеет атрибут ro(только для чтения). Все дополнительные SDO серверы по умолчанию отключены(биты 31 установлнеы), они описываются индексами 1201h-127fh. Недопустимо менять COB-ID, в то время когда SDO создано(invalid_bit ==0). Элемент с подиндексом 3h необязателен для серверов SDO, и недопустим для default SDO(опять же очень странный параметр, а вообще смысл представляется следующим – на сервере можно задавать любую пару ID клиент->сервер и сервер->клиент, не перекрывающую какие то имеющиеся ID, а клиент чтобы получить доступ к этому серверу должен знать их, при чём если сервер устройства должен общаться с несколькими клиентами, то фактически нужно создавать столько серверов SDO, сколько ожидается клиентов.).
ОПИСЕНИЕ ОБЬЕКТА

Index	1200h-127fh
Name	Описатель SDO сервера
Object code	RECORD
Data type	SDO parametr(22h)
Category	для 1200h необязателен(O) для 1201h-127f обязателен для каждого поддерживаемого SDO сервера

ОПИСАНИЕ ЭЛЕМЕНТА.

Sub-index	0h
Description	Количество элементов
Entry category	Обязателен(M)
Access	ro
PDO mapping	No
Value range	для 1200h: 2h для 1201-127fh: 2h-3h
Defaul value	No

Sub-index	1h
Description	COB-ID Client->Server (rx)
Entry category	Обязателен(M)
Access	для 1200h: ro для 1201-127fh: rw
PDO mapping	No
Value range	UNSIGNED32
Defaul value	для 1200h: 600h+Node-ID для 1201-127fh: отключен

Sub-index	2h
Description	COB-ID Server -> Client (tx)
Entry category	Обязателен(M)
Access	для 1200h: ro для 1201-127fh: rw
PDO mapping	No
Value range	UNSIGNED32
Defaul value	для 1200h: 580h+Node-ID для 1201-127fh: отключен

Sub-index	3h
Description	Node-ID SDO клиента
Entry category	Необязателен(O)
Access	rw
PDO mapping	No
Value range	1h-7fh
Defaul value	No

Обьект 1200h-127fh: Описатель SDO клиента.

Эти обьекты описывают SDO клиентов. Если такой элемент поддерживается, то все подэлементы должны присутствовать. Структура этих элементов совпадает со структурой описателей SDO серверов. Все клиенты по умолчанию отключены.

ОПИСЕНИЕ ОБЬЕКТА

Index	1200h-127fh
Name	Описатель SDO сервера
Object code	RECORD
Data type	SDO parametr(22h)
Category	обязателен для каждого поддерживаемого SDO клиента

ОПИСАНИЕ ЭЛЕМЕНТА.

Sub-index	0h
Description	Количество элементов
Entry category	Обязателен(M)
Access	ro
PDO mapping	No
Value range	3
Defaul value	3

Sub-index	1h
Description	COB-ID Client->Server (tx)
Entry category	Обязателен(M)
Access	rw
PDO mapping	No
Value range	UNSIGNED32
Defaul value	отключен

Sub-index	2h
Description	COB-ID Server -> Client (rx)
Entry category	Обязателен(M)
Access	rw
PDO mapping	No
Value range	UNSIGNED32
Defaul value	отключен

Sub-index	3h
Description	Node-ID SDO сервера
Entry category	Обязателен(M)
Access	rw
PDO mapping	No
Value range	1h-7fh
Defaul value	No

Обьект 1400h-15ffh: Параметр связи RPDO.

Содержит параметры связи для тех PDO которые устройство может принимать. Тип параметр связи(20h) описывается в п. 9.5.4. Подиндекс 0h даёт количество действительных элементов в данной записи. Их должно быть как минимум 2. Если поддерживается интервал запрета следующей передачи(время подавления обьекта), то количество элементов 3. В элементе с подиндексом 1 находится COB-ID PDO. Тип данных этого элемента UNSIGNED32, что позволяет использовать как 11битные идетификаторы, так и 29битные.

	UNSIGNED32				
	MSB				LSB
bits	*31*	*30*	*29*	*28-11*	*10-0*
11-bit-ID	0/1	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	11-bit Identifier
29-bit-ID	0/1	0	1	29 -bit Identifier	

Рисунок 65. Структура Идентификатора COB SDO.

Номер бита	Значение	Описание
31(MSB)	0 1	PDO создан/ действителен PDO не создан/ недействителен
30	0 1	Допускается удалённый запрос данного PDO Не допускается удалённый запрос данного PDO
29	0 1	11-битный идентификатор(CAN 2.0A) 29-битный идентификатор(CAN 2.0B)
28-11	0 X	Если бит 29 = 0 Если бит 29 ≠ 0, содержит биты 28-11 29 битного идентификатора
10-0(LSB)	X	биты 10-0 идентификатора

Таблица 55. Описание элемента PDO COB-ID.

Бит 31(флаг наличия) позволяет выбирать PDO, которые будут использоваться в рабочем режиме. Может быть что PDO сконфигурирован, но не используется. Эта особенность применяется в устройствах поддерживающих более чем 4 RPDO и 4 TPDO поскольку каждое устройство имеет предопределённые идентификаторы толко для 4+4 PDO. Устройства поддерживающие стандартный размер CAN-фрейма или не допускающие передачи PDO по удалённому запросу, должны пи попытке установить бит 29 или сбросить бит 30 ответить сообщением обрыва передачи(код 0609 0030h). Не допускается менять биты 0-29 пока обьект действителен(пока бит 31 сброшен). Тип передачи определяет характер приёма-передачи (см. п. 9.2.1.1.). В таблице 56 описано использование этого элемента. При попытке изменить значение на то которое не поддерживается устройство сгенерировать сообщение обрыва передачи(код 0609 0030h).

Тип передачи	Передача				
	циклическая	ациклическая	синхронная	асинхронная	удалённый запрос
0		X	X		
1-240	X		X		
241-251	Зарезервированно				
252			X		X
253				X	X
254				X	
255				X	

Таблица 56. Описание типов передачи.

Синхронные(тип передачи 0-240 и 252) означает что передача PDO будет связана с SYNC обьектами как это описано в п.9.3. Предпочтительно чтобы устройства использовали приём SYNC для осуществления вывода или обновления данных принятых в предшествующих RPDO или обновления данных, которые будут переданы в последующих TPDO. Подробности данного механизма зависят от типа устройств и определяются в профилях устройств, для которых этот механизм применяется.

Асинхронные означает что передача PDO не связана с SYNC обьектами.

Тип передачи 0 означает что PDO будет передан один раз с приходом следующего SYNC обьекта.

Тип передачи 1..240 что PDO будет передаваться синхронно и периодически с периодом равным длительности периода SYNC обьектов умноженным на тип передачи.

RPDO всегда всегда обновляются со следующим после их приёма SYNC обьектом, независимо от типа(если значение 0-240).

Тип 252 и 253 означает что PDO передаются только по удалённому запросу передачи. Тип передачи 252 означает что данные обновляются по приёму SYNC обьекта, передаются по удалённому запросу. Тип 253 означает что данные и обновляются и передаются по удалённому запросу(могут налагаться программные и аппаратные ограничения). Данные типы передачи могут быть только для TPDO.

Тип передачи 254 означает что обмен данными происходит в связи с программным событием определённым изготовителем, тип передачи 255 означает что обмен данными происходит в связи с программным событием определённым профилем устройств. Для RPDO обновление отображаемой информации происходит по приёму RPDO.

Элемент с подиндексом 3h содержит время подавления. Данное время указывает минимальное время между двумя последовательными передачами данного PDO. Единицы параметра сотни микросекунд. Недопустимо менять данный параметр, когда PDO создан(бит 31 по индексу 1h сброшен).

Элемент с подиндексом 4h зарезервирован, и при попытке записать/прочитать этот элемент должно быть произведено аварийное завершение SDO трансферта с кодом 0609 0011h.

В режимах 254/255 для передачи TPDO дополнительно может использоваться таймер событий. Если таймер событий существует(значение периода не равно 0), то переполнение таймера рассмативается как событие, в дополнение к иным событиям, возникновение события устанавливает таймер(в смысле взводит??). Период переполнения таймера расположенный в элементе с подиндексом 5h, задаётся в миллисекундах. Независимо от типа передачи, для RPDO таймер событий используется для определения потери RPDO.

ОПИСЕНИЕ ОБЬЕКТА

Index	1400h-15ffh
Name	Параметр связи RPDO.
Object code	RECORD
Data type	SDO CommPar(20h)
Category	обязателен для каждого поддерживаемого PDO

ОПИСАНИЕ ЭЛЕМЕНТА.

Sub-index	0h
Description	Количество элементов
Entry category	Обязателен(M)
Access	ro
PDO mapping	No
Value range	2-5
Defaul value	

Sub-index	1h
Description	COB-ID данного PDO
Entry category	Обязателен(M)
Access	ro rw – если поддерживаются меняемые значения идентификаторов.
PDO mapping	No
Value range	UNSIGNED32(таблица 55)
Defaul value	PDOс индексом 1400h: 200h + Node-ID PDOс индексом 1401h: 300h + Node-ID PDOс индексом 1402h: 400h + Node-ID PDOс индексом 1403h: 500h + Node-ID PDOс индексом 1404h – 15ffh - запрещены

Sub-index	2h
Description	Тип передачи
Entry category	Обязателен(M)
Access	ro rw – если поддерживаются меняемые типы передачи.
PDO mapping	No
Value range	UNSIGNED8
Defaul value	Определяется профилем устройства

Sub-index	3h
Description	Время подавления
Entry category	Необязателен(O)
Access	rw
PDO mapping	No
Value range	UNSIGNED16
Defaul value	No

Sub-index	4h
Description	Поле для совместимости
Entry category	Необязателен(O)
Access	rw
PDO mapping	No
Value range	UNSIGNED8
Defaul value	No

Sub-index	5h
Description	Таймер событий
Entry category	Необязателен(O)(не используется для RPDO)
Access	rw
PDO mapping	No
Value range	0 – не используется UNSIGNED16
Defaul value	No

Обьект 1600h-17ffh: Параметр отображения RPDO.

Описывает отображение для PDO которые могут приниматься. Тип Параметр отображения PDO (PDO mapping parameter – 21h) описывался в п 9.5.4. Элемент с подиндексом 0h содержит количество элементов отображаемых в PDO, - то есть количество переменных приложений, которые передаются/принимаются с помощью данного PDO. Элементы с индеками 1- фактическое количество элементов(максимум 40h), хранят информацию об отображённых переменных приложений. Формат этих элементов: индекс подиндекс и длина в битах отображаемого элемента, как показано на рисунке 66. Параметр длина отображаемой переменной обязателен, и служит для проверки полной длинны PDO отображения.

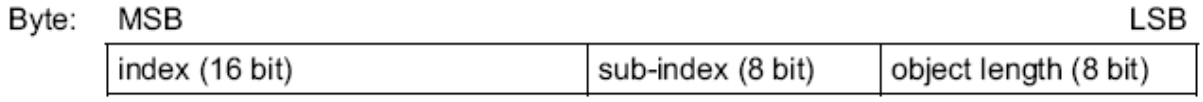


Рисунок 66. Структура описателя отображаемого элемента.

Если не может быть выполнено изменение PDO отображения(например если длина превысила максимально возможную, или SDO клиент пытается отобразить обьект который не может быть отображён), устройство должно произвести аварийное завершение SDO трансферта

Элемент с подиндексом 0h определяет количество элементов, которые могут быть отображены. Для изменения отображения, PDO должен быть сначала удалён(в бит 31 COB-ID записывается 1), затем в элемент с подиндексом 0 должен быть записан 0. После этого обьект может быть переотображён. Когда отображается новый обьект(путём занаесения в элемент отображения его индекса и подиндекса), устройство может проверить существует ли обьект с данным индексом/подиндексом. Если обьект не существет или не может быть отображён, то SDO трансферт должен быть аварийно прерван с кодом ошибки 0602 0000h или 0604 0041h, соответственно. После того как отображены все требуемые обьекты, в элемент параметра отображения с подиндексом 0h записывается действительное количество отображённых в PDO параметров. В конце, PDO должен быть создан путём записывания идентификатора (COB-ID) в параметр связи соответствующего PDO. Когда элемент с подиндексом устанавливается из значения 0 в значение количества действительных элементов отображённых в данном PDO, устройство (SDO сервер) перед квитированием SDO сервиса может проверить вновь заданное PDO отображение. В случае обнаружения ошибки устройство должно произвести Аварийное завершение SDO трансферта с кодом 0602 0000h, 0604 0041h или 0604 0042h.

При чтении элемента с подиндексом 0 получается реальное количество действительных отображённых обьектов.

Если в обьекте отображены типы данных(элементы с индексом 0-7h), то они служат «фиктивными элементами». Данные соответствующие данным параметрам отображения не рассматриваются устройством. Данная дополнительная особенность может использоваться, например, для устройств, которые используют один PDO, и каждое из которых использует только часть PDO. В таком случае для тех полей RPDO, которые не содержат данные для этого устройства, отображение должно указывать на один из таких обьектов - типов данных(1-7h). Нельза создавать фиктивные элементы в TPDO.

Устройства поддерживающие динамическую настройку отображения(то есть настройку отображения во время работы) должны производить его при сетевом статусе устройства PREOPERATIONAL. Если динамическое отображение задаётся в состоянии OPERATIONAL, то SDO клиент который производит это всё, ответственен за цельность и согласованность данных.

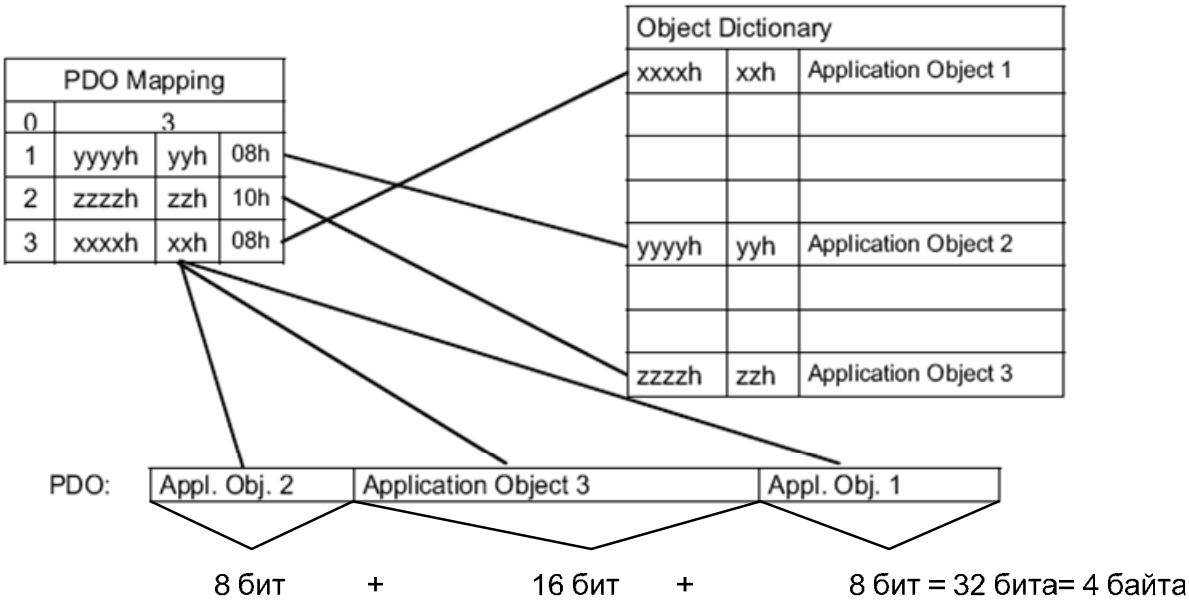


Рисунок 67. Принцип PDO отображения.

ОПИСЕНИЕ ОБЬЕКТА

Index	1600h-17ffh
Name	Параметр отображения RPDO.
Object code	RECORD
Data type	PDO отображение (21h)
Category	обязателен для каждого поддерживаемого PDO

ОПИСАНИЕ ЭЛЕМЕНТА

Sub-index	0h
Description	Количество отображённых в PDO элемнтов приложений.
Entry category	Обязателен(M)
Access	ro rw, если поддерживается динамическая настройка отображения
PDO mapping	No
Value range	0 неактивен 1-64 активен
Defaul value	Определяется профилем устройства

Sub-index	1h – 40h
Description	PDO отображение для Nного прикладного обьекта, который отображён.
Entry category	Обязателен(M) дла каждого отображённого элемента
Access	rw
PDO mapping	No
Value range	UNSIGNED32
Defaul value	Определяется профилем устройства

Обьект 1800h-19ffh: Параметр связи TPDO.

Содержит параметры связи для тех PDO которые устройство может передавать. Тип параметр связи PDO(20h) описывается в п. 9.5.4. Подробное описание данного параметра приводится в описании аналогичного параметра для RPDO(обьект 1400h – 15ffh).

ОПИСЕНИЕ ОБЬЕКТА

Index	1800h-19ffh
Name	Параметр связи TPDO.
Object code	RECORD
Data type	SDO CommPar(20h)
Category	обязателен для каждого поддерживаемого PDO

ОПИСАНИЕ ЭЛЕМЕНТА.

Sub-index	0h
Description	Количество элементов
Entry category	Обязателен(M)
Access	ro
PDO mapping	No
Value range	2-5
Defaul value	

Sub-index	1h
Description	COB-ID данного PDO
Entry category	Обязателен(M)
Access	ro rw – если поддерживаются меняемые значения идентификаторов.
PDO mapping	No
Value range	UNSIGNED32(таблица 55)
Defaul value	PDOс индексом 1800h: 180h + Node-ID PDOс индексом 1801h: 280h + Node-ID PDOс индексом 1802h: 380h + Node-ID PDOс индексом 1803h: 480h + Node-ID PDOс индексом 1804h – 19ffh - запрещены

Sub-index	2h
Description	Тип передачи
Entry category	Обязателен(M)
Access	ro rw – если поддерживаются меняемые типы передачи.
PDO mapping	No
Value range	UNSIGNED8(таблица 56)
Defaul value	Определяется профилем устройства

Sub-index	3h
Description	Время подавления
Entry category	Необязателен(O)
Access	rw
PDO mapping	No
Value range	UNSIGNED16
Defaul value	Определяется профилем устройства

Sub-index	4h
Description	Зарезервировано
Entry category	Необязателен(O)
Access	rw
PDO mapping	No
Value range	UNSIGNED8
Defaul value	No

Sub-index	5h
Description	Таймер событий
Entry category	Необязателен(O)
Access	rw
PDO mapping	No
Value range	0 – не используется UNSIGNED16
Defaul value	No

Обьект 1a00h-1bffh: Параметр отображения TPDO.

Описывает отображение для PDO которые могут передавться. Тип Параметр отображения PDO (PDO mapping parameter – 21h) описывался в п 9.5.4. Подробное описание данного параметра приводится в описании аналогичного параметра для RPDO(обьект 1600h – 17ffh).

ОПИСЕНИЕ ОБЬЕКТА

Index	1a00h-1bffh
Name	Параметр отображения TPDO.
Object code	RECORD
Data type	PDO отображение (21h)
Category	обязателен для каждого поддерживаемого PDO

ОПИСАНИЕ ЭЛЕМЕНТА

Sub-index	0h
Description	Количество отображённых в PDO элемнтов приложений.
Entry category	Обязателен(M)
Access	ro rw, если поддерживается динамическая настройка отображения
PDO mapping	No
Value range	0 неактивен 1-64 активен
Defaul value	Определяется профилем устройства

Sub-index	1h – 40h
Description	PDO отображение для Nного прикладного обьекта, который отображён.
Entry category	Обязателен(M) дла каждого отображённого элемента
Access	rw
PDO mapping	No
Value range	UNSIGNED32
Defaul value	Определяется профилем устройства

10. Рекомендации по реализации

При реализации протоколов следует придерживаться нижеприведённых рекомендаций для гарантирования нормального взаимодействия компонентов сети. Рекомендации касаются следующих аспектов:

Недействительные обьекты связи.

Обьект связи считается недействительным, который имеет идентификатор используемый в соответствии с описанными протоколами, но содержит недопустимое значение параметра согласно с описанием протокола. Это может быть вызвано толко ошибками произошедшими на более нижнем уровне, или ошибками реализации. Недействительный обьект связи должен быть урегулирован **локально**, способом зависящем от особенностей реализации, и который не попадает в рамки данной спецификации. То что касается сетевого протокола, то недействительный обьект связи должен быть проигнорирован.

Тайм-ауты.

Постольку-поскольку обьекты связи могут быть проигнорированы, то на стороне отправившей такой обьект могут никогда не дождаться подтверждения, требуемого таким обьектом. Для разрешения данной счекотливой ситуации, реализация может после некоторого количества времени индицировать таймаут обслуживающемуся пользователю. **Тайм-аут ни в коем случае не может рассматриваться как подтверждение сервиса**(в отличие видимо от аварийного завершения трансферта, который хоть и отрицательный, но всё же результат). Тайм-аут показывает что сервис совсем не завершён. Такие ситуации должно рассматривать приложение. Величины тайм-аутов считаются зависящими от реализации и не попадают в рамки данного документа. Однако рекомендуется, чтобы реализация предоставляла средства для подстройки этих величин в зависимости от требований приложений.