1. Протокол обмена информацией в инструментальных сетях (DiBUS): *Ревизия 11.*

Инструментальная сеть НПП «Доза» (далее просто сеть) имеет мономастерную архитектуру. Ведомые устройства могут выдавать данные в сеть только по запросу «мастера». «Мастер» должен производить соответствующие опросы всех подключенных к нему по сети устройств.

Все пакеты, передаваемые по сети, имеют одинаковую структуру вида:

Заголовок (14 байт)						Блок данных	
Адрес получателя	Адрес отправителя	Тип пакета	Тип данных или интерфейс	Размер данных, байт	CRC* заголовка	Тело блока данных	CRC* блока данных
3 байта	3 байта	1 байт	1 байт	2 байта	4 байта	0..32767 байт	4 байта

* *алгоритм расчёта CRC смотрите см. п.2.3*

«Адрес получателя» и «адрес отправителя» рассматриваются п.1.1.

«Типы пакетов» определены п.1.2.

«Типы данных» определены п.1.3.

«Размер данных» - это количество информации в поле «Тело блока данных», в байтах.

«CRC заголовка» - 4-байтовое число, полученное при обработке первых десяти байтов заголовка алгоритмом, описанным п.2.3.

«Тело блока данных» - собственно данные, определённые типом передаваемого пакета.

«CRC блока данных» - 4-байтовое число, полученное при обработке всех байтов поля «Тело блока данных» алгоритмом, описанным п.2.3. Для пакетов без данных («размер данных» = 0) «CRC блока данных» не рассчитывается и не входит в состав пакета (в этом случае пакет состоит только из заголовка).

Ведомое устройство на любую принятую команду высылает ответный пакет, подтверждая этим приём команды. Форма ответа строго фиксирована для каждого типа запроса *(см. п.1.2)*. В случае, если ведомое устройство не может выслать корректный ответ на команду, оно должно выслать пакет с признаком ошибки *(см. п.2.2)*.

1.1 Адрес получателя, адрес отправителя.

Адрес отправителя – уникальный сетевой адрес (УСА) устройства, отправляющего пакет.

Адрес получателя – УСА получателя или зарезервированный шаблон (см. п.1.1.1).

УСА должен быть уникальной константой, присвоенной устройству на этапе изготовления. Уникальность адресов в устройствах обеспечивается централизованным учётом выданных диапазонов.

Каждому проекту централизовано выделяется некоторый диапазон адресов.

Адрес делится на поля: проект-тип – 1-й байт, тип -2-й байт, серийный номер -3-й байт. При выпуске партии устройств более 253, на тип устройства выделяется следующее число.
Единая таблица диапазонов адресов, зарегистрированных типов и ответственных разработчиков хранится на файловом сервере «Дозы» в каталоге «ID\\Уникальные идентификаторы.xls». В указанном файле прописан также порядок присвоения нового адреса или диапазона адресов.

1.1.1 Широковещательные адреса.

Адреса «0.0.0», «1.1.1» и «255.255.255» зарезервированы за шаблонами, и не могут быть использованы в устройствах.
Адрес «0.0.0» соответствует шаблону адреса незарегистрированного устройства. При приёме пакета с адресом получателя «0.0.0» на него должны ответить все незарегистрированные ведомые устройства. О регистрации см. п. 2.1.
Адрес «255.255.255» соответствует шаблону адреса любого устройства в сети, кроме мастера - отправителя.
Адрес «1.1.1» соответствует адресу устройства «Мастер» в сети.
Шаблонные адреса «0.0.0» и «255.255.255» может указывать в качестве «адреса получателя» только «Мастер». Шаблонный адрес «1.1.1» может указывать в качестве «адреса получателя» только «ведомое устройство».

1.2 Тип пакета

Код	**Команда**	**размер данных, байт**
0	запрос на регистрацию «ведомого»	1 *
1	подтверждение «ведомым» приёма команды	0
2	подтверждение регистрации «ведомого»	1 *
3	Ошибка от устройства	1 ** (код ошибки)
4	запрос «Подключен?» (Ping)	0
5	ответ «ведомого» – хочу передать пакет	2
6	получить данные у «ведомого»	определяется типом данных***
7	запрошенные данные от «ведомого»	определяется типом данных***
8	передать данные «ведомому»	определяется типом данных***
9	Перенаправленный вызов (Redirect)	произвольный****
10	запросить пакет «ведомого»	0
12	разрегистрация «ведомого»	0

Дополнительные типы пакетов будут формироваться по мере необходимости при дополнительном согласовании.
* *о процедуре регистрации смотрите см. п.2.1*
** *коды ошибок представлены п.2.2*
*** *типы данных представлены п.1.3.1*
**** *Команда «Redirect» описана п.1.3.2*

Пакет с кодом **«0»** высылается «мастером» для регистрации ведомых устройств *(см. п. 2.1)*.

Пакет с кодом **«1»** («подтверждение приёма») высылается «ведомым устройством» в ответ на все команды, если они правильно распознаны, исполнены, но не требуют другого вида ответа (на команды «0», «2», «4», «8», «12»).

Пакет с кодом **«2»** высылается «мастером» для регистрации ведомых устройств *(см. п. 2.1)*.

Пакет с кодом **«3»** высылается «ведомым устройством» в том случае, если устройство не способно выполнить команду. «Мастер» высылает пакет с кодом «3» только в определённых случаях использования команды «Redirect» (код 9) (см. п.1.3.2). Ответом «ведомого» устройства должен быть пакет с кодом «1».

Пакет с кодом **«4»** высылается «мастером» для проверки подключения ведомых устройств, а также для опроса устройств, способных к активной передаче пакетов. В ответ на эту команду «ведомое устройство» высылает «5», если хочет передать пакет или «1» в противном случае.

Пакет с кодом **«5»** высылается «ведомым устройством» в ответ на команду «4», если устройство «хочет» самостоятельно выслать какой-либо пакет. При этом в поле «данные» передаётся длина пакета, который хочет передать устройство. В ответ на «5» «Мастер» может выслать запрос «10», для получения пакета ведомого устройства.

Пакет с кодом **«6»** высылается «мастером» для получения информации от ведомых устройств. Способ идентификации данных определяется при этом полем «тип данных» *(см. п. 1.3)*. В ответ «ведомое устройство» должно выслать пакет с кодом «7».

Пакет с кодом **«7»** высылается «ведомым устройством» в качестве ответа на «запрос данных» с кодом «6». Способ идентификации данных определяется при этом полем «тип данных» *(см. п. 1.3)*.

Пакет с кодом **«8»** высылается «мастером» для передачи информации ведомым устройствам. Способ идентификации данных определяется при этом полем «тип данных» *(см. п. 1.3)*. В ответ «ведомое устройство» должно выслать пакет с кодом «1».

Пакет с кодом **«9»** высылается «ведомым устройством» либо «мастером» для передачи информации какому-либо внешнему устройству. При этом в поле «Тип данных или интерфейс» определяется номер интерфейса, на который передаются данные. Работа с перенаправленными пакетами рассматривается п.1.3.2.

Пакет с кодом **«10»** высылается «мастером» в ответ на пакет с кодом «5» для получения информации, которое хочет передать ведомое устройство. В ответ «ведомое устройство» должно выслать пакет с кодом «7» или пакет с кодом «9», если предполагается перенаправление данных.

Пакет с кодом **«12»** высылается «мастером» для разрегистрации «ведомых устройств». В ответ «ведомое устройство» должно изменить свой внутренний статус на «незарегистрированное» и выслать пакет с кодом «1». Данная команда рекомендована к применению для реализации синхронизации статусов регистрации в «мастере» и «ведомых устройствах». Как вариант, предлагается использование нескольких циклов команды «12» с адресом получателя «255.255.255» при старте «мастера».

Команды «0», «1», «2», «4» рекомендованы к реализации во всех сетевых устройствах с доступом «по регистрации».

1.3 Тип данных или интерфейс

1.3.1 Тип данных (используется при передаче данных)

Код	Тип данных	Идентификация
1	**Байт, массив байтов**	**«по индексу»**
2		**«по имени»**
3	**Строка однобайтовых символов**	**«по индексу»**
4		**«по имени»**
5	**Word – беззнаковое целое, 2 байта (младший байт первый)**	**«по индексу»**
6	**(0-65535)**	**«по имени»**
7	**ShortInt*** **– целое со знаком, 1 байт (-128..127)**	**«по индексу»**
8		**«по имени»**
9	**Integer**** **– целое со знаком, 2 байта (младший байт первый)**	**«по индексу»**
10	**(-32768..32767)**	**«по имени»**
11	**DWord – беззнаковое целое, 4 байта (младший байт первый)**	**«по индексу»**
12		**«по имени»**
13	**L_Single – вещественное без знака, 2 байта (xxxxxxyy**	**«по индексу»**
14	**yyyyyyyy – x-порядок, y-мантисса) (младший байт первый)**	**«по имени»**
15	**S_Single – вещественное без знака, 2 байта (xxxxxyyy**	**«по индексу»**
16	**yyyyyyyy – x-порядок, y-мантисса) (младший байт первый)**	**«по имени»**
17	**Массив произвольного типа**	**«по индексу»**
18		**«по имени»**
19	**Непрерывный фрагмент массива произвольного типа**	**«по индексу»**
20		**«по имени»**
21	**ASCII-целое со знаком**	**«по индексу»**
22		**«по имени»**
23	**ASCII-инженерный**	**«по индексу»**
24		**«по имени»**
25	**Single (IEEE-754 Single) – вещественное, 4 байта (младший**	**«по индексу»**
26	**байт (биты 0..7) первый) см. Приложение 1.**	**«по имени»**
27	**M_Single – вещественное, 4 байта**	**«по индексу»**
28	[bit31] **sxxxxxxx xxxxxxxx xxxxxxxx eeeeeeee** [bit0] **s -** знак мантиссы (1 бит) **x -** мантисса (23 бита, нормализация необязательна) **e -** десятичный порядок + 127 (8 бит) младший байт передается первым	**«по имени»**
29	**Строка Unicode (двухбайтовые символы)**	**«по индексу»**
30		**«по имени»**
31	**Long_DateTime – дата и время**	**«по индексу»**
32	[bit63] *yy mm dd hh mm ss msec* [bit0] yy – год (1 байт) mm – месяц (1 байт) dd – число месяца (1 байт) hh – часы (1 байт) mm – минуты (1 байт) ss – секунды (1 байт) msec – миллисекунды (2 байт (слово))	**«по имени»**
33	**DiBUS_address**	**«по индексу»**

34	[bit23] *AA BB CC* [bit0] AA – 1-й байт (проект-тип) (1 байт) BB – 2-й байт (тип) (1 байт) CC – 3-й байт (серийный номер) (1 байт)	**«по имени»**
35	**BCD_Long_DateTime – дата и время в формате BCD**	**«по индексу»**
36	[bit63] *msec ss min hh dd mm yy* [bit0] yy – год (1 байт) mm – месяц (1 байт) dd – число месяца (1 байт) hh – часы (1 байт) min – минуты (1 байт) ss – секунды (1 байт) msec – миллисекунды (2 байта)	**«по имени»**
37	**BCD_BYTE**	**«по индексу»**
38		**«по имени»**
39	**QWord – беззнаковое целое, 8 байт (младший байт первый)**	**«по индексу»**
40		**«по имени»**
125	**Элемент пользовательского типа**	**«по индексу»**
126		**«по имени»**
128	**«Параметрический» запрос / ответ***	

Дополнительные типы данных будут формироваться по мере необходимости при дополнительном согласовании.

* *Отрицательные числа в ShortInt представлены в доп. коде (80h=-1…FFh=-1)*

** *Отрицательные числа в Integer представлены в доп. коде*

*** *Параметрический протокол описан п.2.4*

Базовые типы.

Для типов 1÷24 идентификация переменных производится двумя различными способами.

Идентификация «***по индексу***» предполагает в поле данных один дополнительный байт, идущий перед самими данными, который реализует идею номера переменной. Таким образом, доступом «по индексу» можно именовать до 256 элементов каждого типа.

Идентификация «***по имени***» предполагает в поле данных один дополнительный строковый идентификатор, идущий перед самими данными и являющийся именем переменной. Имя переменной может содержать только буквы английского алфавита, цифры и знак подчёркивания. Максимальный размер строкового идентификатора – 16 байт ASCII-символов, вместе с заканчивающим строку символом #0. Таким образом, доступом «по имени» можно именовать произвольное число элементов каждого типа. Заглавные и прописные буквы строкового идентификатора различаются. Однако строго не рекомендуется называть разные переменные в одной системе идентификаторами, различающимися лишь регистром символов.
Строковый идентификатор должен начинаться с самого начала «данных» и заканчиваться символом #0 (байтовый код 0). Сразу за «#0» должны идти данные.

Подробнее:

Для **«1»** – размер массива = «размеру данных» -1, в случае использования этого типа как массива. При использовании ссылок на этот тип (в типах 17, 18, 19, 20, 125, 126) подразумевается ссылка на байт.

Для **«2»** – размер массива = «размеру данных» - «длина строки идентификатора», в случае использования этого типа как массива. При использовании ссылок на этот тип (в типах 17, 18, 19, 20, 125, 126) подразумевается ссылка на байт.

Для **«3»** и **«4»** – строка однобайтовых символов – это любое количество однобайтовых символов, оканчивающихся однобайтовым символом с кодом 0.

Для **«5»** ÷ **«12»** – типы стандартные для ObjectPascal.

Для **«13»** и **«14»** – L_Single – вещественное без знака, 2 байта (xxxxxxyy yyyyyyyy – x-порядок, y-мантисса) (младший байт первый).

Примеры:
```
6Fh 3Dh = 03D6Fh=00111101 01101111 b=3.67*10^15;
93h F7h = 0F793h=11110111 10010011=9.15*10^-3
```

Для **«15»** и **«16»** – S_Single – вещественное без знака, 2 байта (xxxxxyyy yyyyyyyy – x-порядок, y-мантисса) (младший байт первый)

Для **«17»** и **«18»** - после индекса или имени - идентификатора массива идёт 1 байт номера типа. Этот байт характеризует тип каждого элемента массива. Соответствие значения этого байта типу элементов можно взять из той же таблицы 1.3.1. При этом рекомендуется выбирать нечётные значения типов (элемент выбранного типа «по индексу»). Далее идут значения элементов указанного типа без дополнительных разделителей. Допустимо в качестве типа элементов массива использовать сложные структуры, например записи (тип 125). При этом данные, характеризующие тип записи присутствуют в пакете до начала значений, в единственном экземпляре (см. описание типа «125»).

Пример 1: [массив с индексом 7]:=((1,1), (2,2))
данные – массив структур из двух элементов с типами полей 1,5.

Заголовок – поле «Тип данных или интерфейс»: 11h
Блок данных: 07h 7Dh 02h 01h 05h 01h 01h 00h 02h 02h 00h

Пример 2: [массив с именем “DOSE”]:=((1,1), (2,2));
данные – массив структур из двух элементов с типами полей 5,5.

Заголовок – поле «Тип данных или интерфейс»: 12h
Блок данных: 44h 4fh 53h 45h 00h 7Dh 02h 05h 05h 01h 00h 01h 00h 02h 00h 02h 00h

Для **«19»** – после индекса-идентификатора массива идёт 1 байт номера типа. Этот байт характеризует тип каждого элемента массива. Соответствие значения этого байта типу элементов можно взять из той же таблицы 1.3.1. При этом рекомендуется выбирать нечётные значения типов (элемент выбранного типа «по индексу»). За байтом типа идут дополнительные четыре байта. Первые 2 байта из них содержат начальный индекс массива, вторая пара байт – количество элементов (в паре первый байт - младший). Далее идут значения элементов указанного типа без дополнительных разделителей. Допустимо в качестве типа элементов массива использовать сложные структуры, например записи (тип 125). При этом данные, характеризующие тип записи, присутствуют в пакете до начала значений, в единственном экземпляре.

Индексация элементов в массиве произвольна и определяется пользователем. Рекомендуется присваивать первому элементу массива индекс 0.

> ***Пример:*** [массив с индексом 4]:=(10,11,12,13,14,15,16,17,18,19,20)
> данные – массив элементов типа 5 (WORD) из 11 элементов.
> В пакете передаются эл-ты массива в кол-ве 5 шт, начиная с эл-та с индексом 3.
>
> *Заголовок – поле «Тип данных или интерфейс»*: 13h
> *Блок данных:* 04h 05h 03h 00h 05h 00h 0Dh 00h 0Eh 00h 0Fh 00h 10h 00h 11h 00h

Для **«20»** - после имени идентификатора массива – идёт номер типа элементов массива (так же, как для «19», это может быть номер простого типа или номер типа и описание структуры). За ним следует строковый идентификатор начального элемента, затем идёт строковый идентификатор количества элементов. Формат строк аналогичен типу «21».Далее, как и в «19» - сами элементы.

> ***Пример:*** [массив с именем “DOSE”]:=(10,11,12,13,14,15,16,17,18,19,20)
> данные – массив элементов типа 5 (WORD) из 11 элементов.
> В пакете передаются эл-ты массива в кол-ве 5 шт, начиная с эл-та с индексом 3.
>
> *Заголовок – поле «Тип данных или интерфейс»*: 14h
> *Блок данных:* 44h 4fh 53h 45h 00h 05h 33h 00h 35h 00h 0Dh 00h 0Eh 00h 0Fh 00h 10h 00h 11h 00h

Для **«21»** и **«22»** – ASCII-целое это строка (как в «3») содержащая только ASCII-коды цифр числа с возможным ведущим символом знака. Отсутствие знака тождественно знаку «+».

Примеры:

«45676»	=45676
«-145568»	=-145568
«+7»	=7

Для **«23»** и **«24»** – ASCII-инженерный это строка (как в «3») содержащая только ASCII-коды цифр, символы «.», «E» и символы знака («+», «-») в определённой форме:
[-,+]X.X[X…]E[-,+]Y[Y…]
- в квадратных скобках указаны необязательные элементы;
- «…» обозначает произвольное число указанных элементов;
- «X» - цифра мантиссы;
- «Y» - цифра порядка.

Примеры:

«4.5676E-5»	=0.045676
«-1.4E56»	$=-1.4*10^{56}$
«+7.0E2»	=700.0

Для **«25»** и **«26»** – Single (IEEE-754 Single) – вещественное, 4 байта (младший байт (биты 0..7) первый). Подробнее формат рассмотрен в Приложении 1.

Для **«27»** и **«28»** – M_Single – вещественное, 4 байта (младший байт (биты 0..7) передается первым).
[bit31] **sxxxxxxx xxxxxxxx xxxxxxxx eeeeeeee** [bit0]
- **s -** знак мантиссы (1 бит)
- **x -** мантисса (23 бита, нормализация необязательна)
- **e -** десятичный порядок + 127 (8 бит)

Примеры: `80h 00h 04h 7Eh = -4*10^-1;`
`00h 00h FFh 7Fh = 255`

Для **«29»** и **«30»** – Unicode строка (строка двухбайтовых символов) – это любое количество двухбайтовых символов, оканчивающееся двухбайтовым символом с кодом 0.

Для **«31»** и **«32»** – Long_DateTime – дата и время.
[bit63] *yy mm dd hh mm ss msec* [bit0]

yy – год (1 байт)
mm – месяц (1 байт)
dd – число месяца (1 байт)
hh – часы (1 байт)
mm – минуты (1 байт)
ss – секунды (1 байт)
msec – миллисекунды (2 байт (слово))

Младший байт передается первым.

Для **«33»** и **«34»** – DiBUS_address – адрес устройства в формате DiBUS.
[bit23] *AA BB CC* [bit0]

AA – 1-й байт (проект-тип) (1 байт)
BB – 2-й байт (тип) (1 байт)
CC – 3-й байт (серийный номер) (1 байт)

Младший байт передается первым.

Для **«35»** и **«36»** – QWord – целое без знака, 8 байт (младший байт первый).

Для **«125»** и **«126»** – после индекса или имени - идентификатора пользовательского типа, идёт один байт – размер «заголовка записи» (количество полей в записи). «Заголовок записи» - это последовательность байт, характеризующих тип каждого элемента записи. Соответствие значений этих байт типам элементов можно взять из той же таблицы 1.3.1. При этом рекомендуется выбирать нечётные значения типов. Далее идут значения элементов указанного типа без дополнительных разделителей. После «заголовка записи» следует значение записи в виде значений, характерных для каждого элемента, расположенных далее без каких-либо разделителей в соответствии с указанными типами.

Пример: структура, состоящая из 3 полей с типами 5,1,7.
данные: элемент массива с индексом 1. Поля элемента (структуры) равны 1 (WORD), 2 (BYTE), 0 (SHORTINT)

Заголовок – поле «Тип данных или интерфейс»: 7Dh
Блок данных: 01h 03h 05h 01h 07h 01h 00h 02h 00h 00h

Этот пример описывается на языке паскаль следующим образом:

```
type
    TData = packed record
      Value1: Word;
      Value2: Byte;
      Value3: ShotInt;
     end;
var
   Val: array [0..255] of TData;
   ...
```

Val[1]:= (1,2,0);

Для **«128»** – данные определяют протокол текстового обмена информацией, описанный далее п.2.4.

При запросах данных (команда «6»), в пакете присутствует только идентификатор или индекс переменной. При возврате / установке значений (команды «7», «8»,), значения переменных следуют сразу за их идентификатором / индексом.
В командах, которые не используют поле «тип данных или интерфейс» (команды «0», «1», «2», «3», «4», «5», «10», «12»), значение этого поля зарезервировано и должно быть равно «0».

1.3.2 Пакет «Redirect» содержит в своём теле пакет, адресованный другому устройству.

<table>
<tr><td>Заголовок</td><td>CRC</td><td colspan="4">Данные</td><td>CRC</td></tr>
<tr><td></td><td></td><td>Заголовок</td><td>CRC</td><td>Данные</td><td>CRC</td><td></td></tr>
<tr><td></td><td></td><td colspan="4">Любой формат пакета, воспринимаемый драйвером интерфейса</td><td></td></tr>
</table>

При этом заголовок включает в себя номер интерфейса

Интерфейс (используется при передаче пакетов «Redirect») (тип пакета «9»)

Код	**Интерфейс***
0	**Этот же интерфейс**
1-254	**Определяется разработчиком устройства**
255	**Все доступные интерфейсы**

Дополнительные типы данных будут формироваться по мере необходимости при дополнительном согласовании.

* *Интерфейс - это устройство, обеспечивающее канал передачи данных (последовательный порт, сетевая карта или любое другое).*

«Ведомое устройство», поддерживающее команду “Redirect”, при приёме соответствующего пакета, должно выслать на интерфейс под номером, указанным в байте «интерфейс» пакет, содержащийся в теле. При этом запоминается интерфейс и адрес, с которого пришёл запрос. Когда на посланный пакет по указанному интерфейсу приходит ответ, этот ответ отправляется на запомненный адрес в теле данных в виде нового пакета «Redirect». В случаях, когда ответа нет больше максимально возможного срока ожидания для данного интерфейса, рекомендуется очистить запомненный интерфейс с адресом, не высылая предупреждений. При заведомо известном возможном превышении времени ответа с запрошенного интерфейса, следует использовать механизм активного отправления пакетов (комбинация команд «4», «5», «10», «9»). Подобным образом можно передавать данные через «ведомое устройство» как через шлюз в другие варианты сетей на другой физической основе. В случае если «Redirect» на указанный интерфейс не поддерживается, или драйвер указанного интерфейса находит ошибку структуры, «ведомое устройство» высылает «мастеру» признак ошибки «10» *(см. п.2.2)*.

«Мастер», поддерживающий команду “Redirect”, при приёме соответствующего пакета, должен выслать на интерфейс под номером указанном в байте «интерфейс» пакет, содержащийся в теле. При этом запоминается интерфейс и адрес, с которого пришёл запрос. Когда на посланный пакет по указанному интерфейсу приходит ответ, этот ответ отправляется на запомненный адрес в теле данных в виде нового пакета «Redirect». В случаях, когда ответа нет больше максимально возможного срока ожидания для данного интерфейса, - рекомендуется очистить запомненный интерфейс с

адресом, не высылая предупреждений. Подобным образом «ведомое устройство» может передавать данные через «мастер» как через шлюз в другие варианты сетей на другой физической основе. В случае если «Redirect» на указанный интерфейс не поддерживается, или драйвер указанного интерфейса находит ошибку структуры, «мастер» высылает «ведомому устройству» признак ошибки «10» *(см. п.2.2)*.

2. Приложения.

2.1 Регистрация

2.1.1 Полноценная регистрация.

Для полноценной работы в сети, любое устройство должно зарегистрироваться. Для «горячей» регистрации сетевых устройств, устройство «мастер» должно периодически делать запрос с кодом «0» по адресу «0.0.0». В «данных» этого запроса передаётся любое меняющееся число (например, номер запроса). При этом любое устройство, подключенное к сети, но не зарегистрированное «мастером», должно выдать в ответ пакет с кодом «1» и с адресом отправителя, установленном при изготовлении. Задержка перед отправкой этого пакета определена п.2.1.3. «Мастер» должен подождать 256*24t, пока не придут все возможные ответы от устройств. Затем «мастер» регистрирует те устройства, которые сочтёт необходимым, высылая им подтверждение командой «2». При этом он в качестве данных посылает «параметр задержки» от 2 до 255. Этот параметр должен быть различным во всех подключенных устройствах. Если предполагается широковещательный опрос каких-либо данных, следует выбирать значения «задержек» таким образом, чтобы не возникало коллизий из-за длительности передачи этих данных. «Мастер» считает регистрацию оконченной, когда в ответ приходит подтверждение приёма.

2.1.2 Упрощённая регистрация.

Упрощённый вариант регистрации – регистрация «холодная». При прямом обращении к любому «ведомому устройству», оно должно считать себя зарегистрированным. Параметр «задержки» при этом остаётся неинициализированным до тех пор, пока «мастером» не будет прислана команда «2» по прямому адресу. В этом случае, не следует использовать широковещательных команд до «раздачи» параметров «задержки».

Механизм упрощённой регистрации может быть использован совместно с регистрацией полноценной.

2.1.3 Рекомендуемый алгоритм расчёта задержки при полноценной регистрации.

Для адреса устройства A.B.C, и псевдослучайного числа X, пришедших при запросе на регистрацию, задержку рекомендуется рассчитывать по следующему алгоритму:

Задержка = (lo(A*X) xor lo(B*X*2) xor lo(C*X*4) *25 + 1)*24t,

где t-время передачи 1-го байта,

lo – младший байт.

2.2 Коды ошибок

Код	Ошибка
1	Неподдерживаемая команда
2	Неподдерживаемый формат данных
3	Некорректная структура пакета
4	Нет указанной переменной
5	Устройство занято и не может ответить сейчас
6	Устройство занято, ответ будет выслан по готовности…
7	CRC заголовка корректно, CRC данных некорректно
10	Некорректный пакет «Redirect»
255	Нераспознанная ошибка

Ошибка **«1»** выдаётся «ведомым устройством» в случае приёма им команды, которую оно не поддерживает.

Ошибка **«2»** выдаётся «ведомым устройством» в случае приёма им команды, с типом данных, которые им не поддерживаются.
Ошибка **«3»** выдаётся «ведомым устройством» в случае приёма им команды, с нарушениями в структуре пакета.
Ошибка **«4»** выдаётся «ведомым устройством» в случае приёма им команды, с идентификатором, которые им не поддерживаются.
Ошибка **«5»** выдаётся «ведомым устройством» в случае приёма им команды, на которую оно не может дать ответа в течение отведённого регламента времени.
Ошибка **«6»** выдаётся «ведомым устройством» в случае приёма им команды, на которую оно даст ответ позже «самостоятельно».
Ошибка **«7»** выдаётся «ведомым устройством» в случае приёма им пакета, в котором контрольная сумма не соответствует принятым данным, что может произойти, например, из-за импульсной помехи при передаче длинного пакета.
Ошибка **«10»** выдаётся «ведомым устройством» или «мастером» в случае приёма им некорректного содержимого команды «Redirect».
Ошибка **«255»** выдаётся «ведомым устройством» в случае приёма принципиальной невозможности реализовать корректный разбор ошибки. Не рекомендуется к применению.

2.3 CRC

В простых микроконтроллерах чрезвычайно сложно реализовать стандартные алгоритмы CRC. Расчёт нашего варианта CRC основывается на комбинации арифметического и логического сложений со сдвигом.
За основу взят алгоритм XOR32.
Ретрансляция из ассемблера в ObjectPascal Delphi имеет вид:

```
function CalculateCRC(Data: array of byte) : DWORD;
  var
    CRC         : DWORD;
    DataSize,
    i           : LongInt;
begin
  CRC := 0;
  i := Low(Data);
  DataSize := High(Data) - Low(Data) + 1;  // SizeOf(Data)
  if DataSize mod 2 = 1 then begin
    CRC := CRC xor DWord(Data[i]);
    Inc(i);
  end;
  while i < DataSize do begin
    CRC := Rol(CRC,5);
    CRC := CRC xor ((DWORD(Data[i]) shl 8) + DWORD(Data[i+1]));
    Inc(i, 2);
  end;
  CalculateCRC := CRC;
end;
```

2.4 Параметрический протокол

Блок данных (с учетом контрольной суммы транспортного уровня)

Поле	Размер поля	Положение в пакете (номера байтов)
Индекс блока данных - 0 – блок единственный и полный - 1..255 – порядковый номер фрагмента полного блока данных (блок фрагментирован)	1 byte	0

Вид блока данных - 0 – блок с набором параметров (1) - 1 – блок с непрерывным дампом чего-нибудь	1 byte	1
Тело блока данных	n-2 byte	2..n-1
CRC	4 byte	n..n+3

(1) – набор параметров представляет собой последовательность ASCII-конструкций вида:

```
<ID_параметра>[:<атрибут_параметра>]=<значение_параметра>;
```

где

`<ID_параметра>`	идентификатор параметра
`<атрибут_параметра>`	атрибут параметра (необязательный, является расширением идентификатора параметра)
`<значение_параметра>`	1. значение параметра - в случае корректного ответа на запрос; 2. @@ - значение параметра недоступно (параметр не поддерживается абонентом). 3. ?? - запрос значения параметра в ASCII-виде; 4. ??#□ - запрос значения параметра в виде дампа : ??## - запрос количества фрагментов в дампе, ответ в виде `<ID_параметра>[:<атрибут_параметра>]=##n;` где n=1..255 (кол-во фрагментов) ??#n - запрос фрагмента № n, n=1..255

2.5 Активная посылка пакетов от ведомых устройств.

Команда с кодом «5» высылается «ведомым устройством» в ответ на команду «мастера» с кодом «4». «Мастер» соответственно должен производить опрос подключённых устройств этой командой.

В команде с кодом «5», «ведомое устройство» определяет размер блока данных, который хочет передать. После отсылки команды с кодом «5», «ведомое устройство» не должно менять данные, предназначенные для отправки. Исключение составляют однотипные измерения, которые должны быть актуальны на момент передачи.

Активная посылка пакетов «ведомыми устройствами» удобна для реализации событийной логики.

3. Физический уровень

3.1 Рекомендуемый интерфейс:

Рекомендуемый интерфейс RS-485, полудуплекс, 1 стоповый бит без контроля чётности.

Для семейства RS-232, RS-485, RS-422:

Основная скорость обмена 9600 бод (t – время передачи 1-го байта = 1мс). Устройства, в которых реализована поддержка других скоростей обмена, должны также поддерживать основную скорость 9600 бод и обеспечивать удобный механизм выбора скорости обмена и установки основной скорости.

3.2 Временные характеристики интерфейса:

Задержка между байтами в процессе передачи пакета не более 3*t.
Минимально допустимая задержка между пакетами = 6*t.

3.3 Временные задержки при ответах ведомыми устройствами:

Устройство, подключенное к сети, должно отвечать на все команды, посылаемые ему по прямому адресу (не широковещательные) – не ранее чем через 6*t после приёма последнего байта запроса. Максимальное время задержки ответа составляет при этом 40*t.

Если устройству необходимо произвести некоторые дополнительные действия перед ответом и величина задержки при производимых действиях может быть больше 40*t, оно должно выслать ответ с признаком ошибки «5» или «6». Ошибка 5 – «Устройство занято и не может ответить сейчас» высылается, если устройство не планирует ответить позже самостоятельно, в противном случае высылается Ошибка 6 – «Устройство занято, ответ будет выслан по готовности…» *(см. п.2.5)*.

При приёме зарегистрированным сетевым устройством пакета с адресом «255.255.255», оно должно выслать ответ с задержкой, определённой при регистрации в пакете подтверждения регистрации (код 2). Величина задержки должна быть равна «Параметр задержки»*24*t, где t-время передачи 1-го байта. «Параметр задержки» передается ведомому устройству Мастером в качестве параметра команды подтверждения регистрации (код 2) *(см. п. 3.2)*.
При приёме незарегистрированным сетевым устройством пакета с адресом «255.255.255», оно должно выслать ответ с задержкой от 6*t до 40*t.
При приёме незарегистрированным сетевым устройством команды «запрос на регистрацию» (код 0) с адресом «0.0.0» задержка ответа рассчитывается по формуле, описанной в 2.1.
При приёме незарегистрированным устройством любого другого пакета с адресом «0.0.0» ответ высылается с произвольной псевдослучайной задержкой в пределах (1-255)*24t, где t-время передачи 1-го байта. Механизм генерации псевдослучайной задержки должен исключать многократное повторение одних и тех же чисел в различных устройствах.
Если устройство не может генерировать псевдослучайные числа, обмен с ним осуществляется только по прямому адресу или после регистрации. На все другие команды (кроме запроса на регистрацию) по адресу «0.0.0» оно не должно отвечать.
Выбор параметра задержки «мастером» при подтверждении регистрации (код 2) *(см. п.2.1)* осуществляется из принципа несовместимости одних времён задержек в разных устройствах сети. Кроме того, этот параметр не должен быть равен «0» или «1».

Приложение 1. Форматы чисел.

IEEE-754. Single Format

The IEEE single format consists of three fields: a 23-bit fraction, f; an 8-bit biased exponent, e; and a 1-bit sign, s. These fields are stored contiguously in one 32-bit word, as shown in Figure 2-1. Bits 0:22 contain the 23-bit fraction, f, with bit 0 being the least significant bit of the fraction and bit 22 being the most significant; bits 23:30 contain the 8-bit biased exponent, e, with bit 23 being the least significant bit of the biased exponent and bit 30 being the most significant; and the highest-order bit 31 contains the sign bit, s.

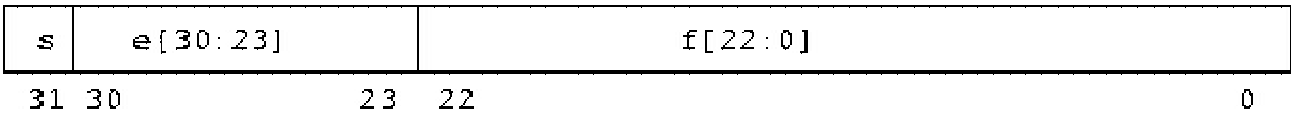

Figure 2-1 Single-Storage Format

Table 2-2 shows the correspondence between the values of the three constituent fields s, e and f, on the one hand, and the value represented by the single- format bit pattern on the other; *u* means *don't care*, that is, the value of the indicated field is irrelevant to the determination of the value of the particular bit patterns in single format.

Table 2-2 Values Represented by Bit Patterns in IEEE Single Format

Single-Format Bit Pattern	Value
$0 < e < 255$	$(-1)^s \times 2^{e-127} \times 1.f$ (normal numbers)
$e = 0$; $f \neq 0$ (at least one bit in f is nonzero)	$(-1)^s \times 2^{-126} \times 0.f$ (subnormal numbers)
$e = 0$; $f = 0$ (all bits in f are zero)	$(-1)^s \times 0.0$ (signed zero)
$s = 0$; $e = 255$; $f = 0$ (all bits in f are zero)	+INF (positive infinity)
$s = 1$; $e = 255$; $f = 0$ (all bits in f are zero)	-INF (negative infinity)
$s = u$; $e = 255$; $f \neq 0$ (at least one bit in f is nonzero)	NaN (Not-a-Number)

of figures shown Notice that when $e < 255$, the value assigned to the single format bit pattern is formed by inserting the binary radix point immediately to the left of the fraction's most significant bit, and inserting an implicit bit immediately to the left of the binary point, thus representing in binary positional notation a mixed number (whole number plus fraction, wherein 0 <= fraction < 1).
The mixed number thus formed is called the *single-format significand*. The implicit bit is so named because its value is not explicitly given in the single- format bit pattern, but is implied by the value of the biased exponent field.
For the single format, the difference between a normal number and a subnormal number is that the leading bit of the significand (the bit to left of the binary point) of a normal number is 1, whereas the

leading bit of the significand of a subnormal number is 0. Single-format subnormal numbers were called single-format denormalized numbers in IEEE Standard 754.
The 23-bit fraction combined with the implicit leading significand bit provides 24 bits of precision in single-format normal numbers.
Examples of important bit patterns in the single-storage format are shown in Table 2-3. The maximum positive normal number is the largest finite number representable in IEEE single format. The minimum positive subnormal number is the smallest positive number representable in IEEE single format. The minimum positive normal number is often referred to as the underflow threshold. (The decimal values for the maximum and minimum normal and subnormal numbers are approximate; they are correct to the number.)

Table 2-3 Bit Patterns in Single-Storage Format and their IEEE Values

Common Name	Bit Pattern (Hex)	Decimal Value
+ 0	00000000	0.0
- 0	80000000	-0.0
1	3f800000	1.0
2	40000000	2.0
maximum normal number	7f7fffff	3.40282347e+38
minimum positive normal number	00800000	1.17549435e-38
maximum subnormal number	007fffff	1.17549421e-38
minimum positive subnormal number	00000001	1.40129846e-45
+ ∞	7f800000	Infinity
- ∞	ff800000	-Infinity
Not-a-Number	7fc00000	NaN

A NaN (Not a Number) can be represented with any of the many bit patterns that satisfy the definition of a NaN. The hex value of the NaN shown in Table 2-3 is just one of the many bit patterns that can be used to represent a NaN.