3VS | SOFTWARE DEVELOPMENT

# Спецификация открытого протокола обмена NordWind Open UDP Версия 2.3

Руководитель разработки
Технический директор

_____ Баум Ф.И.
«_____»_____2024 г.

Москва 2024

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Для обмена данными между программным обеспечением NordWind и узлами вычислительной сети существует универсальный открытый протокол обмена данными NordWind Open UDP. Данный протокол разработан с использованием сетевого стека UDP/IP.

Протокол удовлетворяет ряду требований:

1) Позволяет передавать типы данных, используемых в программном обеспечении NordWind, статусы сигналов и времена их возникновения.
2) Гарантирует целостность данных при сетевом обмене.
3) Позволяет диагностировать ошибки использования протокола и потери данных по сети.
4) У протокола отсутствуют ограничения на его использование со стороны Posix – совместимых операционных систем и типов архитектур процессоров участников обмена данными, естественным ограничением может выступать только объем доступной оперативной памяти;
5) Адаптирует скорость обмена данными к реальным характеристикам линии передачи данных с целью уменьшения их потерь.
6) Протокол состоит из двух групп функций:
   - использующих при работе предварительно созданный канал обмена данными (рекомендуется использовать при высокой пропускной способности сети);
   - используемых при низкой пропускной способности сети, без создания канала обмена данными.
7) Реализация клиентской части протокола в виде структур данных и функций на языке СИ с открытым исходным кодом;

Протокол предназначен для обмена данными между разными сегментами автоматизированных систем управления технологическими процессами (АСУ ТП). Для использования протокола NordWind Open UDP разработчик стороннего программного обеспечения должен подключить библиотеку функций, написанных на языке программирования C.

При использовании протокола разработчик стороннего программного обеспечения может произвести модификацию функций протокола в зависимости от тех задач, которые необходимо решить.

Протокол реализован на языке программирования C.

Протокол NordWind Open UDP является универсальным протоколом, и его применение не ограничивается только использованием при создании и эксплуатации АСУ ТП с использованием программного обеспечения SimInTech и программного обеспечения NordWind, протокол также может быть применен при создании АСУ ТП, в состав которых входят компоненты, реализованные без использования SimInTech и NordWind на сторонних аппаратных и программных платформах.

# 1 ОПИСАНИЕ ПРИНЦИПОВ РАБОТЫ ПРОТОКОЛА

## 1.1 Обмен данными с предварительно созданным каналом обмена данными

В случае если характеристики сети не ниже 100 кбит/с, рекомендованы к использованию группа функций, использующих в своей работе предварительно созданный канал обмена. Под каналом обмена будем понимать создание на стороне клиента и сервера структур содержащей характеристики одноименной группы сигналов, между которыми будет происходить обмен данными («Сессию обмена данными»).

Для организации канала обмена данными необходимо создать структуру «GROUP», в которой данные, полученные по сети от программного обеспечения NordWind, хранятся в упорядоченном виде для выполнения дальнейших операций с этими данными. Структура «GROUP»

Содержимое структуры «GROUP» приведено ниже (Рисунок 1).

```
struct GROUP
{
    int                *indx; /*Массив индексов сигналов*/
    int                *len;   /*Размер значения сигнала*/
    unsigned char   *data_type; /*Тип данных 0-вещественный, 1-двоичный, 2-целый*/
    int                 num;   /*Количество сигналов в группе*/
    int                 size;  /*Размер значений для группы сигналов*/
    unsigned char      *value;  /*Хранятся значения в байтовом потоке*/
    int                *status; /*Массив статусов сигнала*/
    double              *time;  /*Массив времен возникновения синалов*/
    int                 *offset; /*Хранятся смещения*/

};
```

Рисунок 1 – Структура данных в структуре «GROUP»

Структура «GROUP» хранит значения, статусы, времена возникновения сигналов, а также информацию для выполнения операций с данными группы.

Протокол подразумевает использование двух групп данных:

1) группа на стороне клиента (удаленная группа);
2) группа на стороне сервера обмена NordWind_UDP_Server (локальная» группа).

Локальная группа получает данные из памяти (базы данных) программного обеспечения NordWind.

Под обменом данных подразумевается обмен данными между локальной и удаленной группами (удаленная группа является проекцией локальной группы). Для работы с группой существуют функции, которые позволяют оперировать данными из нее: через использование индекса может происходить получение или запись соответствующих данных. Поэтому получение и запись данных в память NordWind будут реализованы через группу.

При работе с программным обеспечением NordWind протоколом предусмотрено наличие NordWind_UDP_Server. NordWind_UDP_Server реализован в виде UDP – сервера, который имеет доступ к области разделяемой памяти программного обеспечения

NordWind: для обмена данными с клиентом будет создаваться структура «GROUP» (при этом список сигналов как на стороне сервера, так и на стороне клиента одинаков) и через нее будет происходить обмен данными с shared memory программного обеспечения NordWind с использованием функций NordWind SDK.

Для использования протокола обмена данными необходимо:

1. Со стороны клиента инициализировать соединение с сервером.

2. Открыть «Сессию обмена данными», для этого необходимо создать канал обмена данными между клиентом и сервером. В результате на стороне клиента и сервера будет создана однотипная структура описывающая список сигналов обмена. В случае ошибки обмен данными не возможен и для начала обмена данными необходимо добиться положительного выполнения команды. Команда для создания канала обмена данными выполняется один раз, в случае если сервер был перезапущен канал обмена данными необходимо создать еще раз.
3. После открытия «Сессии обмена данными» возможно использовать функции протокола для реализации обмена данными описание функций см. ниже. При использовании функций обмена стоит анализировать код возврата функций в случае успешно выполненной операции 0x001, в случае потери связи или не совпадение контрольных сумм отрицательное значение. В случае аварийного завершения работы функций по коду возврата необходимо установить причину ошибки в случае если для клиента не создан канал обмена данными, необходимо его создать и повторно использовать функцию.
4. Сервер может последовательно выполнять команды от нескольких клиентов открывших «Сессии обмена данными» с сервером.
5. «Сессии обмена данными» завершена если от клиента получена команда на закрытие в результате на стороне сервера будет освобождена память от структуры со списком сигналов обмена «GROUP» и для продолжения обмена необходимо заново открыть «Сессии обмена данными».

На стороне сервера для каждого клиента создается группа и заносится в хэш-очередь по IP-адресу. Для каждого клиента (при этом подразумевается, что каждый клиент имеет свой IP-адрес) создается только одна группа.

При отправке команд со стороны клиента сервер сначала производит проверку того, создана ли для этого клиента группа или нет, а затем по результатам проверки возможны следующие варианты:

1) если группа существует, то команда будет выполнена;
2) если группа не существует, то будет выдан код ошибки.

В случае отправки повторной команды на создание группы предыдущая группа будет удалена, и будет создана новая группа для нового списка сигналов.

От клиента можно получить шесть команд (cmd=1...6).

Для выполнения команд первое обращение клиента должно быть с командой cmd=1, которая подразумевает создание группы для приема переданного по сети списка сигналов. В случае удачной реализации команд клиент получит необходимы данные или код возврата, в случае положительного выполнения команд код возврата больше нуля, а в случае ошибок отрицательный.

Протокол выполняет следующее определение ошибок:

- отсутствие запрашиваемого сигнала при создании канала обмена;
- таймаут;
- неверные контрольные суммы;
- отсутствие созданной «Сессии обмена данными» для клиента;
- ошибка передачи по сети.

Стоит обратить внимание!

При обмене данными между клиентом и сервером размер буфера данных может превышать 1400 Байт, поэтому при посылке данных в сеть по протоколу UDP буфер данных будет разбит на отдельные пакеты размером 1400 Байт, таким образом мы гарантируем отсутствие фрагментации пакета и можем быть уверены в целостности данных внутри каждого пакета (протокол UDP использует анализ контрольной суммы при приеме/передачи пакетов с данными по сети). При передачи данных по UDP возможны потери пакетов, для оценки целостности данных используется расчет контрольной суммы CRC32 для всего переедаемого/принимаемого буфера данных, таким образом после завершения приема/передачи можно оценить целостность данных и в случае отсутствия таковой повторить обмен.

## 1.2 Обмен данными без создания канала обмена данными

Данную группу функций целесообразно использовать при условии существования сети с низкой пропускной способностью (~10 Кбод) и отсутствие необходимости получать значения статусов и времен возникновения сигналов. Предварительно необходимо подготовить три файла («discrets.dat», «analogs.dat», «int.dat») со списками имен дискретных, аналоговых и целочисленных сигналов соответственно, именно с сигналами из перечисленных файлов будет происходить обмен данными. При реализации обмена данными с использованием описываемой группы функций данные передаются в составе одного пакета по протоколу UDP размер которого не превышает 1400 Байт, дефрагментации пакета не происходит, а использование контрольной суммы протоколом UDP позволяет быть уверенным в целостности данных. Адресация данных будет определяться порядком нахождения имен сигналов в соответствующих файлах. Вне зависимости от расположения сигналов в базе данных NordWind. Файлы со списком сигналов должны быть расположены в рабочей директории сервера NordWind_UDP_Server, при старте сервера если хоть один из сигналов не будет найден в базе данных NordWind сервер прекращает свою работу с соответствующей диагностикой. Для использования протокола необходимо:

1. Со стороны клиента инициализировать соединение с сервером.
2. При использовании функций протокола анализировать коды возврата и в случае отрицательных кодов возврата использовать функции повторно.
3. В случае перезапуска сервера нет необходимости в повторном пере инициализации соединения

Протокол выполняет следующее определение ошибок:

- Потерю пакетов с данными;

– Неверную структуру данных;
– Ошибки сети;

# 2 ОПИСАНИЕ ФУНКЦИЙ ПРОТОКОЛА ИСПОЛЬЗУЮЩИХ КАНАЛ ОБМЕНА ДАННЫМИ

В протоколе NordWind Open UDP реализованы функции для работы с данными с предварительно созданным каналом обмена. В данном разделе приведен список функций, с помощью которых реализован обмен данными между удаленной и локальной группами на стороне клиента и сервера соответственно.

Функции расположены в файле «nw_exchange.c», заголовки в файле «nw_exchange.h». Данные файлы предоставляются вместе с файлами, описывающими протокол NordWind Open UDP, при запросе к разработчику протокола.

1.
```
int Init_NW_Exchange_Udp_Protocol(char *ip_addr, uint16_t port, int *socket_udp);
```

Обмен данными происходит с использованием протокола UDP.

В результате вызова этой функции создается присоединённый сокет, что необходимо для того, чтобы в пакете всегда был адрес отправителя (если сокет не присоединен, то адрес может меняться), по адресу клиента сервер определяет группу данных, для которой пришел запрос от клиента.

2.
```
int Do_Remote_Groupe(char (*name)[64], uint32_t num, int fd_sock, struct GROUP *gr)
```

При использовании данной функции создается группа данных на стороне клиента, а затем аналогичная группа создается на стороне сервера для данного IP-адреса.

При повторном использовании этой функции существующая группа будет удалена: в случае, если группа данных уже есть, будет создана новая группа данных для нового списка сигналов. Предварительно необходимо выделить память для структуры «GROUP».

В результате выполнения функции на стороне клиента будет получена инициализированная группа для заданного списка сигналов, и в момент инициализации группа будут иметь такие же значения, как и локальная группа на стороне UDP-сервера.

Обмен данными между клиентом и сервером приведен в таблицах 1 и 2.

Таблица 1 – Создание локальной группы на стороне сервера и данные, посылаемые клиентом

| Описание | Размер | Значения |
|---|---|---|
| Команда серверу на создание группы сигналов для обмена с клиентом cmd | 1 Байт | 0x001 |
| Размер пакета передаваемого клиентом по сети nbytes | Байт | Целое число |
| Количество сигналов в группе num | uint32_t | Целое число |
| Контрольная сумма crc32 передаваемого буфера | uint32_t | Целое число |
| Длина имени сигнала len | uint32_t | Целое число |
| Имя сигнала name | char[64] | Массив байт |
| Массив пар {len, name} | …… | ……. |
| Длина имени сигнала len[num] | uint32_t | Целое число |
| Имя сигнала name[num] | char[64] | Массив байт |

Таблица 2 – Ответ сервера на запрос клиента

| Описание | Размер | Значения |
|---|---|---|
| Размер пакета посланного сервером клиенту nbytes | Байт | Целое число |
| Количество сигналов в группе num | Байт | Целое число |
| Размер байт необходимы для хранения значений сигналов группы size_of_value | Байт | Целое число |
| Значение контрольной суммы информационной части сигнала crc32 | uint32_t | Целое число |
| Массив значений размеров сигналов `len[Num_Signal_Packet]`, где `Num_Signal_Packet` количество сигналов в группе | uint32_t размер элемента массива | Массив целых чисел. Если сигнала отсутствует в базе данный (shm) NW, то значение 1 Байт |
| Массив значений типов данных сигналов data_type[`Num_Signal_Packet`], где `Num_Signal_Packet` количество сигналов в группе | uint32_t размер элемента массива | Массив целых чисел. Если сигнала отсутствует в базе данный (shm) NW, то значение 255 |
| Массив байтов значений сигналов values. Значения сигналов хранятся в байтовом | uint8_t | Массив целых чисел. Если сигнала отсутствует в базе данный (shm) NW, то под значение сигнала |

| Описание | Размер | Значения |
|---|---|---|
| массиве в не зависимости от их типа | | выделяется 1Байт равный 0 |
| Массив статусов сигналов status[Num_Signal_Packet], где Num_Signal_Packet количество сигналов в группе | int32_t | Массив целых чисел. Если сигнала отсутствует в базе данный (shm) NW, то значение -1 |
| Массив времен возникновения сигналов time[Num_Signal_Packet], где Num_Signal_Packet количество сигналов в группе | double | Массив вещественных чисел. Если сигнала отсутствует в базе данный (shm) NW, то время возникновение сигнала 0 |

Коды ошибок функции:
1 группа создана;
-1 ошибка сети, тайм-аут;
-2 неверная контрольная сумма.

3.
```
int Update_Remote_Groupe (int fd_sock, struct GROUP *gr)
```

С помощью данной функции происходит синхронизация локальной группы с удаленной группой на стороне UDP-сервера. При синхронизации будут получен значения, статусы, времена возникновения сигналов для всех сигналов, входящих в группу.

Для получения значений сигналов из группы необходимо воспользоваться функцией «get_value_from_group ()», при этом производить обращение по номеру сигнала в группе.

Для выполнения команды обновления группы клиент посылает серверу cmd = 0x03, результат ответа от сервера приведен в таблице 3.

Таблица 3 – Ответ сервера на запрос клиента cmd=0x03

| Описание | Размер | Значения |
|---|---|---|
| Размер пакета посылаемого сервером клиенту nbytes | uint32_t | Целое число |
| Контрольная сумма, последующей части пакета crc32 | uint32_t | Целое число |
| Массив значений сигналов в виде массива Байт value[] | uint8_t | Массив байт |
| Массив статусов сигналов в виде массива байт | uint8_t | Массив байт |
| Массив времен возникновения сигналов в виде массива байт time[] | uint8_t | Массив байт |

Коды ошибок функции:
1 синхронизация произошла;
-1 ошибка сети, тайм-аут;
-2 группа не создана для клиента;
-3 не совпадение контрольной суммы.

4.

```
int Destroy_Group(int fd_sock, struct GROUP *gr);
```

Функция необходима для завершения обмена данными и освобождения ресурсов как на стороне сервера, так и на стороне клиента. В результате работы данной функции будут освобождены структуры данных, приведенные на рисунке 1.

Для реализации данного функционала происходит отправка клиентом команды cmd=0x02.

Коды ошибок функции:

1 группа удалена;

-1 ошибка сети, тайм-аут;

-2 группа не создана для клиента;

5.

```
int Write_Gr_in_NW(int fd_sock, struct GROUP *gr)
```

Обновляем значения локальной группы на стороне UDP-сервера, после чего данные значений сигналов, статусы, времена возникновения записываются в область разделяемой памяти NW. Мы имеем возможность менять данные в группе на стороне клиента используя функцию «set_value_in_group()» перед использованием функции «Write_Gr_in_NW()», но после вызова данной функции все данные группы будут переданы по сети серверу, даже если они не менялись! Формат данных, посланного клиентом серверу (Таблица 4).

Таблица 4 – Запрос клиента

| Описание | Размер | Значения |
|---|---|---|
| Команда серверу на обновление группы сигналов, вся группа будет записана в память NW | 1 Байт | 0x004 |
| Размер пакета посылаемого сервером клиенту nbytes | uint32_t | Целое число |
| Контрольная сумма, последующей части пакета crc32 | uint32_t | Целое число |
| Массив значений сигналов в виде массива Байт value[] | uint8_t | Массив байт |
| Массив статусов сигналов в виде массива байт | uint8_t | Массив байт |
| Массив времен возникновения сигналов в виде массива байт time[] | uint8_t | Массив байт |

Коды ошибок функции:

1 группа удалена;

-1 ошибка сети, тайм-аут;

-2 группа не создана для клиента;

6.

```
Read_Data_From_NW(int fd_sock, struct GROUP *gr, int indx0, int nval)
```

Используя эту функцию, мы запрашиваем не все значения группы, а только заданные (indx0 индекс в группе сигнала, nval количество сигналов начиная с indx0) и по сети будет передан пакет только с определенными нами сигналами. Перед формированием пакета UDP сервер обновит данные группы. Используя функцию «get_value_from_group ()» можно будет получить значения полученных сигналов. Формат данных, посланного клиентом серверу (Таблица 5), ответ сервера (Таблица 6).

Таблица 5 – Запрос клиента на чтение сигналов

| Описание | Размер | Значения |
|---|---|---|
| Команда серверу на чтение сигналов из БД NW в группу cmd | 1 Байт | 0x005 |
| Начальный индекс сигнала в группе indx0 | uint32_t | Целое число |
| Количество считанных сигналов nval | uint32_t | Целое число |

Таблица 6 – Ответ сервера, посылает заданные сигналы

| Описание | Размер | Значения |
|---|---|---|
| Размер пакета посылаемого сервером клиенту nbytes | uint32_t | Целое число |
| Контрольная сумма, последующей части пакета crc32 | uint32_t | Целое число |
| Массив значений сигналов в виде массива Байт value[] | uint8_t | Массив байт |
| Массив статусов сигналов в виде массива байт | uint8_t | Массив байт |
| Массив времен возникновения сигналов в виде массива байт time[] | uint8_t | Массив байт |

Коды ошибок функции:
1 обновление произошло;
-1 ошибка сети, тайм-аут;
-2 группа не создана для клиента;
-3 не совпадение контрольной суммы.

Со стороны сервера код ошибки, см описание функции!

7.

```
int Write_Data_To_NW(int fd_sock, struct GROUP *gr, int indx0, int nval)
```

Используя эту функцию, мы выбираем из группы сигналы и посылаем их UDP серверу для того, чтобы он обновил их в локальной группе. После успешного обновления сигналов в локальной группе сервер записывает значения только обновленных сигналов из группы в память NW. В данном варианте по сети идет пакет с определенными на запись сигналами! Изменить сигналы в группе перед отправкой можно используя функцию «set_value_in_group ()». Формат данных, посланного клиентом серверу (Таблица 7).

Таблица 7 – клиент посылает пакет для записи значений в локальную группу и потом в базе данных NordWind

| Описание | Размер | Значения |
|---|---|---|
| Команда серверу на запись сигналов из локальной группы в БД NW cmd | 1 Байт | 0x006 |
| Размер пакета nbytes | uint32_t | Целое число |
| Начальный индекс сигнала в группе indx0 | uint32_t | Целое число |
| Количество сигналов nval | uint32_t | Целое число |
| Контрольная сумма, последующей части пакета crc32 | uint32_t | Целое число |
| Массив значений сигналов в виде массива Байт value[] | uint8_t | Массив байт |
| Массив статусов сигналов в виде массива байт | uint8_t | Массив байт |
| Массив времен возникновения сигналов в виде массива байт time[] | uint8_t | Массив байт |

Коды ошибок функции:
1 обновление произошло;
-1 ошибка сети, тайм-аут;
-2 группа не создана для клиента;

# 3 ОПИСАНИЕ ФУНКЦИЙ ПРОТОКОЛА НА ОСНОВЕ РАСПОЛАГАЕМОГО СПИСКА СИГНАЛОВ

Данную группу функций целесообразно использовать при условии существования сети с низкой пропускной способностью (~10 Кбод) и отсутствие необходимости получать значения статусов и времен возникновения сигналов. Предварительно необходимо подготовить три файла («discrets.dat», «analogs.dat», «int.dat») со списками имен дискретных, аналоговых и целочисленных сигналов соответственно, именно с сигналами из перечисленных файлов будет происходить обмен данными. При реализации обмена данными с использованием описываемой группы функций данные передаются в составе

одного пакета по протоколу UDP размер которого не превышает 1400 Байт, дефрагментации пакета не происходит, а использование контрольной суммы протоколом UDP позволяет быть уверенным в целостности данных.

8.

```
int Read_BitsData_From_DB_NW (int fd_sock, int indx0, int nval, uint8_t
*bit_signals)
```

Используем данную функцию для чтения значений дискретных сигналов из базы данных (памяти) NW. Индекс indx0 соответствует номеру сигнала в файле «discrets.dat», nval количество сигналов в файле «discrets.dat» начиная с indx0, значения которых мы хотим получить из базы данных NW, bit_signals массив значений дискретных сигналов размерностью nval куда будут транслированы значения дискретных сигналов, запрошенных от удаленного сервера доступа к БД NW. Существует ограничения на количество запрашиваемых сигналов nval не должно превышать 11176 сигналов, это требование сформулировано исходя из необходимости разместить данные в пакете UDP размером 1400 Байт. В таблице 8 представлен запрос клиента серверу доступа к БД NW, а в таблице 9 ответ сервера.

Таблица 8 – Запрос клиента на чтение сигналов

| Описание | Размер | Значения |
|---|---|---|
| Команда серверу на чтение сигналов из БД NW cmd | 1 Байт | 0x007 |
| Начальный индекс сигнала в файле списка имен сигналов indx0 | uint32_t | Целое число |
| Количество считанных сигналов nval | uint32_t | Целое число |
| Случайное число `rand_mean = (rand() % 100);` | uint8_t | Целое число от 0 до 99 |

Таблица 9 – Ответ сервера, посылает заданные сигналы

| Описание | Размер | Значения |
|---|---|---|
| Двух байтовое число 0xFFFF | 2 Байт | 0xFFFF |
| Случайное число rand_mean полученное сервером при запросе от клиента см. таб.8 | uint8_t | Целое число |
| Битовый поток значений дискретных сигналов размером nval | Битовый поток размерностью nval | Массив байт |

В случае ошибки функцией будет возращено значение меньше нуля, в случае успешного выполнения положительное значение.

Коды ошибок функции:

1 данные пришли;

-1 ошибка сети, тайм-аут;

-2 размер запрошенных сигналов превышает допустимое значение;

-3 потеря пакета;

9.

```
int Write_BitsData_To_DB_NW (int fd_sock, int indx0, int nval, uint8_t
*bit_signals)
```

Используем данную функцию для обновления значений дискретных сигналов в базе данных (памяти) NW. Индекс indx0 соответствует номеру сигнала в файле «discrets.dat», nval количество сигналов в файле «discrets.dat» начиная с indx0, значения которых мы хотим обновить в базе данных NW, bit_signals массив значений дискретных сигналов размерностью nval которые будут транслированы серверу доступа к БД NW для перезаписи новых значений сигналов в БД NW. Существует ограничения на количество обновленных сигналов nval не должно превышать 11120 сигналов, это требование сформулировано исходя из необходимости разместить данные в пакете UDP размером 1400 Байт. В таблице 10 представлен запрос клиента серверу доступа к БД NW, а в таблице 11 ответ сервера.

Таблица 10 – Запрос клиента на чтение сигналов

| Описание | Размер | Значения |
|---|---|---|
| Команда серверу на чтение сигналов из БД NW cmd | 1 Байт | 0x0A |
| Начальный индекс сигнала в файле списка имен сигналов indx0 | uint32_t | Целое число |
| Количество считанных сигналов nval | uint32_t | Целое число |
| Случайное число `rand_mean = (rand() % 100);` | uint8_t | Целое число от 0 до 99 |
| Битовый поток значений дискретных сигналов размером nval | Битовый поток размерностью nval | Массив байт |

Таблица 11 – Ответ сервера, посылает заданные сигналы

| Описание | Размер | Значения |
|---|---|---|
| Двух байтовое число 0xFFFF | 2 Байт | 0xFFFF |
| Случайное число rand_mean полученное сервером при запросе от клиента см. таб.10 | uint8_t | Целое число |
| Код возврата | uint8_t | Целое число |

В случае ошибки функцией будет возращено значение меньше нуля, в случае успешного выполнения положительное значение.

Коды ошибок функции:

1 данные обновлены;

-1 ошибка сети, тайм-аут;
-2 размер запрошенных сигналов превышает допустимое значение;
-3 потеря пакета;

10.

```
int Read_IntData_From_DB_NW (int fd_sock, int indx0, int nval, int32_t
*int_signals)
```

Используем данную функцию для чтения значений целых сигналов из базы данных (памяти) NW. Индекс indx0 соответствует номеру сигнала в файле «discrets.dat», nval количество сигналов в файле «int.dat» начиная с indx0, значения которых мы хотим получить из базы данных NW, int_signals массив значений целых сигналов размерностью nval куда будут транслированы значения дискретных сигналов, запрошенных от удаленного сервера доступа к БД NW. Существует ограничения на количество запрашиваемых сигналов nval не должно превышать 349 сигналов, это требование сформулировано исходя из необходимости разместить данные в пакете UDP размером 1400 Байт. В таблице 12 представлен запрос клиента серверу доступа к БД NW, а в таблице 13 ответ сервера.

Таблица 12 – Запрос клиента на чтение сигналов

| Описание | Размер | Значения |
|---|---|---|
| Команда серверу на чтение сигналов из БД NW cmd | 1 Байт | 0x008 |
| Начальный индекс сигнала в файле списка имен сигналов indx0 | uint32_t | Целое число |
| Количество считанных сигналов nval | uint32_t | Целое число |
| Случайное число `rand_mean = (rand() % 100);` | uint8_t | Целое число от 0 до 99 |

Таблица 13 – Ответ сервера

| Описание | Размер | Значения |
|---|---|---|
| Двух байтовое число 0xFFFF | 2 Байт | 0xFFFF |
| Случайное число rand_mean полученное сервером при запросе от клиента см. таб.12 | uint8_t | Целое число |
| Массив значений целых сигналов размером nval | Массив int32_t размерностью nval | Массив int32_t [nval] |

В случае ошибки функцией будет возвращено значение меньше нуля, в случае успешного выполнения положительное значение.

Коды ошибок функции:
1 данные пришли;
-1 ошибка сети, тайм-аут;
-2 размер запрошенных сигналов превышает допустимое значение;

-3 потеря пакета;

11.

```
int Write_IntData_To_DB_NW (int fd_sock, int indx0, int nval, int32_t
*bit_signals)
```

Используем данную функцию для обновления значений целых сигналов в базе данных (памяти) NW. Индекс indx0 соответствует номеру сигнала в файле «discrets.dat», nval количество сигналов в файле «int.dat» начиная с indx0, значения которых мы хотим обновить в базе данных NW, int_signals массив значений целых сигналов размерностью nval которые будут транслированы серверу доступа к БД NW для перезаписи новых значений сигналов в БД NW. Существует ограничения на количество обновленных сигналов nval не должно превышать 347 сигналов, это требование сформулировано исходя из необходимости разместить данные в пакете UDP размером 1400 Байт. В таблице 14 представлен запрос клиента серверу доступа к БД NW, а в таблице 15 ответ сервера.

Таблица 14 – Запрос клиента на чтение сигналов

| Описание | Размер | Значения |
|---|---|---|
| Команда серверу на чтение сигналов из БД NW cmd | 1 Байт | 0x0B |
| Начальный индекс сигнала в файле списка имен сигналов indx0 | uint32_t | Целое число |
| Количество считанных сигналов nval | uint32_t | Целое число |
| Случайное число `rand_mean = (rand() % 100);` | uint8_t | Целое число от 0 до 99 |
| Массив значений целых сигналов размером nval | Массив int32_t размерностью nval | Массив int32_t [nval] |

Таблица 15 – Ответ сервера

| Описание | Размер | Значения |
|---|---|---|
| Двух байтовое число 0xFFFF | 2 Байт | 0xFFFF |
| Случайное число rand_mean полученное сервером при запросе от клиента см. таб.14 | uint8_t | Целое число |
| Код возврата | uint8_t | Целое число |

В случае ошибки функцией будет возращено значение меньше нуля, в случае успешного выполнения положительное значение.

Коды ошибок функции:

1 данные обновлены;

-1 ошибка сети, тайм-аут;

-2 размер запрошенных сигналов превышает допустимое значение;

-3 потеря пакета;

12.

```
int Read_DoubleData_From_DB_NW (int fd_sock, int indx0, int nval, double
*double_signals)
```

Используем данную функцию для чтения значений вещественных сигналов из базы данных (памяти) NW. Индекс indx0 соответствует номеру сигнала в файле «discrets.dat», nval количество сигналов в файле «analogs.dat» начиная с indx0, значения которых мы хотим получить из базы данных NW, double_signals массив значений вещественных сигналов размерностью nval куда будут транслированы значения вещественных сигналов, запрошенных от удаленного сервера доступа к БД NW. Существует ограничения на количество запрашиваемых сигналов nval не должно превышать 174 сигналов, это требование сформулировано исходя из необходимости разместить данные в пакете UDP размером 1400 Байт. В таблице 16 представлен запрос клиента серверу доступа к БД NW, а в таблице 17 ответ сервера.

Таблица 16 – Запрос клиента на чтение сигналов

| Описание | Размер | Значения |
|---|---|---|
| Команда серверу на чтение сигналов из БД NW cmd | 1 Байт | 0x009 |
| Начальный индекс сигнала в файле списка имен сигналов indx0 | uint32_t | Целое число |
| Количество считанных сигналов nval | uint32_t | Целое число |
| Случайное число `rand_mean = (rand() % 100);` | uint8_t | Целое число от 0 до 99 |

Таблица 17 – Ответ сервера, посылает заданные сигналы

| Описание | Размер | Значения |
|---|---|---|
| Двух байтовое число 0xFFFF | 2 Байт | 0xFFFF |
| Случайное число rand_mean полученное сервером при запросе от клиента см. таб.16 | uint8_t | Целое число |
| Массив значений вещественных сигналов размером nval | Массив double размерностью nval | Массив double [nval] |

В случае ошибки функцией будет возращено значение меньше нуля, в случае успешного выполнения положительное значение.

Коды ошибок функции:

1 данные пришли;

-1 ошибка сети, тайм-аут;

-2 размер запрошенных сигналов превышает допустимое значение;

-3 потеря пакета;

13.

```
int Write_DoubleData_To_DB_NW (int fd_sock, int indx0, int nval, double
*double_signals)
```

Используем данную функцию для обновления значений вещественных сигналов в базе данных (памяти) NW. Индекс indx0 соответствует номеру сигнала в файле «discrets.dat», nval количество сигналов в файле «int.dat» начиная с indx0, значения которых мы хотим обновить в базе данных NW, int_signals массив значений вещественных сигналов размерностью nval которые будут транслированы серверу доступа к БД NW для перезаписи новых значений сигналов в БД NW. Существует ограничения на количество обновленных сигналов nval не должно превышать 173 сигналов, это требование сформулировано исходя из необходимости разместить данные в пакете UDP размером 1400 Байт. В таблице 18 представлен запрос клиента серверу доступа к БД NW, а в таблице 19 ответ сервера.

Таблица 18 – Запрос клиента на чтение сигналов

| Описание | Размер | Значения |
|---|---|---|
| Команда серверу на чтение сигналов из БД NW cmd | 1 Байт | 0x0A |
| Начальный индекс сигнала в файле списка имен сигналов indx0 | uint32_t | Целое число |
| Количество считанных сигналов nval | uint32_t | Целое число |
| Случайное число `rand_mean = (rand() % 100);` | uint8_t | Целое число от 0 до 99 |
| Массив значений вещественных сигналов размером nval | Массив double размерностью nval | Массив doublet [nval] |

Таблица 19 – Ответ сервера

| Описание | Размер | Значения |
|---|---|---|
| Двух байтовое число 0xFFFF | 2 Байт | 0xFFFF |
| Случайное число rand_mean полученное сервером при запросе от клиента см. таб.18 | uint8_t | Целое число |
| Код возврата | uint8_t | Целое число |

В случае ошибки функцией будет возращено значение меньше нуля, в случае успешного выполнения положительное значение.

Коды ошибок функции:

1 данные обновлены;

-1 ошибка сети, тайм-аут;

-2 размер запрошенных сигналов превышает допустимое значение;

-3 потеря пакета;

# 4 АДАПТАЦИЯ БИБЛИОТЕКИ ПРОТОКОЛА К ПРОПУСКНЫМ ВОЗМОЖНОСТЯМ КАНАЛА СВЯЗИ

В случае низкой пропускной способности сети для уменьшения вероятности потери данных существует возможность изменять скорость посылки пакетов с данными по протоколу UDP, для этого необходимо изменить время «сна» после посылки пакета в сеть, значение времени сна (в микросекундах) присвоено константе «TIME_DELTA» в файле «nw_exchange.h».