|  |
| --- |
| ООО Эко-Е |
| Описание подсистемы RPC ECO-E |
| Детальное описание реализации подсистемы RPC используемой в устройствах ООО Эко-Е. Внутренняя документация. |
| Автор: Громов Всеволод Владимирович (gromov.vsevolod@yandex.ru) |
| Ревизия: 4 |

Оглавление

[1. Общее описание. 2](#_Toc287859493)

[2. Структура данных RPC. 2](#_Toc287859494)

[2.1. RPC запрос (то, что идет за заголовком пакета): 2](#_Toc287859495)

[2.2. RPC ответ (то, что идет за заголовком пакета). 2](#_Toc287859496)

[3. Типы данных RPC и их размещение в стеке. 2](#_Toc287859497)

[4. Сигнатуры RPC 3](#_Toc287859498)

[5. Стандартные идентификаторы RPC 4](#_Toc287859499)

[6. Идентификатор устройства 4](#_Toc287859500)

[7. Статус выполнения RPC 5](#_Toc287859501)

[8. RPC, реализованные в фотоголовке ФГ-01 5](#_Toc287859502)

# Общее описание.

Механизм RPC реализован поверх нашего протокола, хотя какой именно протокол используется – не имеет значения. Код протокольной команды RPC = 0x20. Ответ 0x8000 + 0x20. Все данные RPC пересылаются как пакет данных протокола.

Максимальная длина пакета не должна превышать 512 байт (заголовок+данные+crc16)

# Структура данных RPC.

## RPC запрос (то, что идет за заголовком пакета):

UINT16 id; // RPC id – число, по которому клиент ищет конкретную реализацию функции в своей lookup таблице. Разные клиенты могут реализовывать стандартные служебные RPC (с идентификаторами от 0 до 2047 включительно), и нестандартные, реализующие собственно функционал клиента, с идентификаторами от 2048 до 65535.

UINT16 sig; // RPC signature – число, определяющее сигнатуру вызова RPC, а именно, тип возвращаемого значения, количество, и типы передаваемых параметров. (Текущая имплементация поддерживает до 6 ти передаваемых параметров.)

RPC stack[RPC\_FRAME\_SIZE-ProtocolEkosfHdr\_SZE-6]; // rpc stack space

Передаваемые параметры размещаются в стеке слева направо, простые типы упаковываются максимально плотно, без выравнивания, весь стек как единое целое может выравниваться по длине на величину кратную 16 бит, добавлением пустого байта (0) в конец стека.

За стеком RPC размещается слово CRC16, согласно спецификации протокола.

## RPC ответ (то, что идет за заголовком пакета).

UINT16 id; // RPC id – идентификатор вызванной процедуры, на которую приходит ответ.

UINT16 status; // RPC status – в случае успешно выполненного вызова RPC равно 0, в противном случае содержит код ошибки, возвращенный клиентом.

RPC stack[RPC\_FRAME\_SIZE-ProtocolEkosfHdr\_SZE-6]; // rpc stack space

Если вызов RPC предполагает возвращение значения, оно помещается в стек. Стек в целом может выравниваться на длину кратную 2, добавлением пустого (0) байта. За стеком RPC, или, в случае, когда возвращение значение не предусмотрено, непосредственно за словом статуса следует CRC16 пакета данных протокола.

# Типы данных RPC и их размещение в стеке.

Подсистема RPC использует следующие простые типы данных:

* INT8;
* UINT8;
* INT16;
* UINT16;
* INT32;
* UINT32;
* INT64;
* UINT64;
* BOOL;
* DATETIME;
* TIMESPAN;
* FLOAT;
* DOUBLE.

Примечания:

* DATETIME и TIMESPAN - это генерализованная реализация календарного типа данных, в качестве внутреннего представления использующего 64-битный счетчик миллисекунд, прошедших с опорной даты, определенной в данной имплементации, как 1.1.2000.
* Тип BOOL упаковывается в стек как 1 байт. 0 = FALSE, 1 = TRUE.

Для передачи произвольных массивов данных (структур) используется специализированный тип BYTEARRAY, описанный как:

typedef struct {

UINT32 size;

BYTE\* data;

} BYTEARRAY;

BYTEARRAY передается в стеке RPC в прямой последовательности, вначале 4 байта - размер данных, затем сами данные.

# Сигнатуры RPC

На настоящий момент определены следующие сигнатуры вызовов RPC:

| Вызов | Значение сигнатуры |
| --- | --- |
| void(void) | 1 |
| UINT8(void) | 2 |
| UINT16(void) | 3 |
| UINT32(void) | 4 |
| UINT64(void) | 5 |
| INT8(void) | 6 |
| INT16(void) | 7 |
| INT32(void) | 8 |
| INT64(void) | 9 |
| BOOL(void) | 10 |
| DATETIME(void) | 11 |
| BYTEARRAY(void) | 12 |
| FLOAT(void) | 13 |
| void(BYTEARRAY) | 14 |
| void(UINT16) | 15 |
| void(FLOAT) | 16 |
| BOOL(DATETIME) | 17 |
| BYTEARRAY(UINT16) | 18 |
| void(UINT8) | 19 |
| void(UINT32) | 20 |
| void(BOOL) | 21 |
| void(UINT8, FLOAT) | 22 |
| FLOAT(UINT8) | 23 |

# Стандартные идентификаторы RPC

|  |  |  |
| --- | --- | --- |
| Идентификатор | Сигнатура вызова | Описание |
| 0 | BYTEARRAY(void) | Возвращает идентификаторы всех поддерживаемых клиентом RPC, в виде массива UINT16 |
| 1 | BYTEARRAY(void) | Возвращает простой или расширенный идентификатор устройства.  Для измерительных головок – простой, для БОИ – расширенный. |
| 4 | UINT32(void) | Возвращает битовую маску состояния устройства. Значения битов в маске могут интерпретироваться по-разному, в зависимости от типа устройства. |
| 5 | void(void) | Инициирует выполнение процедуры самотестирования устройства. В результате выполнения процедуры битовая маска состояния устройства может поменяться. |
| 6 | BOOL(DATETIME) | Устанавливает дату и время в устройстве. Возвращает TRUE, если установка времени была успешна. |
| 7 | DATETIME(void) | Возвращает текущую дату и время устройства |
| 8 | void(void) | Производит сброс настроек устройства к заводским установкам |
| 9 | void(void) | Производит выключение устройства. |
| 10 | void(UINT32) | Переводит устройство в режим загрузчика прошивки, с автоматической перезагрузкой через указанное количество миллисекунд. |
| 11 | BYTEARRAY(void) | Запрашивает статус питания устройства, возвращаемый в структуре PowerStatus |
| 12 |  |  |
| 13 |  |  |
| 14 |  |  |
| 15 |  |  |
| 16 |  |  |
| 17 |  |  |
| 18 |  |  |
| 19 |  |  |
| 20 | void(void) | Запрашивает аудио/визуальную идентификацию устройства |

# Идентификатор устройства

Простой (в основном планируется для измерительных головок):

typedef struct {

UINT16, // тип устройства, уникальное значение от 0 до 65535, 0 и 1 уже зарезервированы

{

UINT16, // год производства

UINT16 // порядковый номер устройства данного типа, произведенного в пределах года

},

{

BYTE, // версия прошивки

BYTE // ревизия прошивки

}

}

Расширенный идентификатор (БОИ) содержит все поля простого, дополнительно включает 32 байтовое поле UID – unique ID, позволяющее осуществить привязку устройства к пользователю, фактически, это строчка GUID без разделителей и скобок.

# Статус выполнения RPC

|  |  |
| --- | --- |
| Код статуса | Описание |
| 0 | RPC успешно завершен |
| 1 | RPC с указанным id не определена на стороне клиента |
| 2 | Стек RPC поврежден (либо клиент не может нормально распаковать стек с параметрами, либо сервер не может нормально распаковать возвращаемый результат из стека) |
| 3 | Выполнение RPC завершено досрочно, из-за превышения времени ожидания ответа клиента на сервере. Если ответ клиента придет позже, он будет проигнорирован сервером. |
| 4 | Переполнение стека. Чаще всего как результат того, что данные превышают максимально допустимую длину (позволенную протоколом). |
| 5 | Неизвестная сигнатура вызова. Чаще всего результат нестыковок прошивок/ПО, построенных в разное время, и содержащих различные описания сигнатур (неконсистентные изменения). |
| 6 | Несоответствие сигнатур серверного запроса и клиентской таблицы RPC. |
| 7 | Идентификатор RPC клиентского ответа не совпадает с идентификатором RPC серверного запроса |
| 8 | Исполнение RPC прервано из-за ошибок коммуникации. Этот код выставляется сервером, зависит от конкретной реализации сервера. |
| 9 | Выполнение RPC не может быть завершено из-за того, что клиент занят, либо запрашиваемые данные не готовы. Конкретная интерпретация этого кода зависит от клиентской реализации. |
| 10-16 | Ошибка значения параметра 0..5. Зависит от реализации конкретной функции. |

# RPC, реализованные в фотоголовке ФГ-01

|  |  |  |
| --- | --- | --- |
| Код | Сигнатура | Описание |
| Стандартные – все указанные в таблице «Стандартные идентификаторы RPC» | | |
| 2048 | void(UINT8) | rpcSetAutosaveInterval – устанавливает интервал автосохранения данных в автономном режиме работы, в секундах. Допустимый интервал значений [1..30] сек. |
| 2049 | UINT8(void) | rpcGetAutosaveInterval – возвращает текущий интервал автосохранения |
| 2050 | void(FLOAT) | rpcSetLuxCorrection – устанавливает калибровочный коэфф.т. для канала lux |
| 2051 | FLOAT(void) | rpcGetLuxCorrection – возвращает текущий калибровочный коэфф.т. для канала lux |
| 2052 | void(FLOAT) | rpcSetCdmCorrection – устанавливает калибровочный коэфф.т. для канала cd/m2 |
| 2053 | FLOAT(void) | rpcGetCdmCorrection – возвращает текущий калибровочный коэфф.т. для канала cd/m2 |
| 2054 | void(FLOAT) | rpcSetKpCorrection – устанавливает калибровочный коэфф.т. для канала Kp |
| 2055 | FLOAT(void) | rpcGetKpCorrection – возвращает текущий калибровочный коэфф.т. для канала Kp |
| 2056 | void(UINT8) | rpcSetOperationMode – устанавливает текущий режим работы:  1 = Idle, 2 = Live – измерение, данные не сохраняются, 3 = Autonomous – измерение, данные сохраняются. |
| 2057 | UINT8(void) | rpcGetOperationMode – возвращает текущий режим работы |
| 2058 | BYTEARRAY(void) | rpcGetLiveData – возвращает текущий результат измерений, в Live или Autonomous режиме |
| 2059 | UINT16(void) | rpcGetDataCount – возвращает текущее количество данных сохраненных в памяти ФГ |
| 2060 | BYTEARRAY(UINT16) | rpcGetDataAtIdx – возвращает данные, сохраненные в памяти под указанным индексом |
| 2061 | BOOL(void) | rpcClearData – очищает массив сохраненных данных. Возвращает TRUE в случае успешного выполнения |
| 2062 | UINT8(void) | rpcGetRange – возвращает текущий диапазон измерений датчика освещенности; 0 – нижний, 1- верхний. |
| 2063 | void(UINT8) | rpcSetRange – устанавливает диапазон измерений фотодатчика |
| 2064 | BOOL(void) | rpcGetAutoRange – возвращает TRUE если активирован режим автопереключения диапазонов фотодатчика (по умолчанию, ФГ всегда стартует в режиме автопереключения диапазонов) |
| 2065 | void(BOOL) | rpcSetAutoRange – включает или выключает режим автопереключения диапазонов фотодатчика |
| 2066 | BOOL(void) | rpcIsLinearLightSensor – возвращает TRUE, если в ФГ используется линейный датчик, FALSE, если логарифмический |