



**Библиотека
Movicom Modbus Library
Версия 0.1**

Руководство пользователя

Movicom, 2009
Rev. 0.1
02.06.2009



Оглавление

Об этом документе.....	3
1. Как начать работать с библиотекой.....	3
1.1. Файлы библиотеки.....	3
1.2. Подключение библиотеки.....	3
1.3. Минимальный код.....	3
2. Устройство библиотеки.....	5
2.1. Карта параметров устройства.....	6
2.1.1. Название карты параметров.....	6
2.1.2. Файлы карты параметров.....	6
2.2. Типы, используемые в библиотеке.....	6
2.2.1. Тип RequestResult (Таблица результатов запроса параметров).....	7
2.2.2. Тип ModbusDeviceID.....	8
2.2.3. Типы ModbusChannelHandle и DeviceHandle.....	8
2.3. Функции библиотеки.....	8
2.3.1. Функция ModbusLibraryInit.....	8
2.3.2. Функция CreateChannel.....	8
2.3.3. Функция DestroyChannel.....	8
2.3.4. Функция ConnectToDevice.....	9
2.3.5. Функция CheckDeviceDefinition.....	9
2.3.6. Функция SetDeviceTimeout.....	10
2.3.7. Функции ReadDeviceParamAsXXX.....	10
2.3.8. Функции WriteDeviceParamAsXXX.....	10



Об этом документе

Данный документ описывает API библиотеки Movicom Modbus Library версии 0.1

1. Как начать работать с библиотекой

Этот раздел описывает краткие шаги о том, как написать собственную программу для работы с библиотекой.

1.1. Файлы библиотеки

Файлы библиотеки, необходимые для работы:

- MovicomModbusLibrary.h – файл API библиотеки
- MovicomModbusLibrary.lib – файл для статической линковки
- MovicomModbusLibrary.dll – динамическая библиотека

Файлы, которые необходимы для работы с устройствами:

Требуется наличие файлов .h и .movilib для устройства, которое вы хотите настраивать. Названия этих файлов отражают к какому устройству они относятся. Так, например, если у вас устройство SampleDevice версии 2.00 с программным обеспечением версии 1.00, следует искать файлы

- SampleDevice.2.00.1.00.h
- SampleDevice.2.00.1.00.movilib

1.2. Подключение библиотеки

Для успешной работы подключите в свою программу файлы .h для библиотеки и для устройства.

1.3. Минимальный код

```
#include "../ModbusLibrary.h"
#include "SampleDevice.2.00.1.00.h"

void PrintConnectError(int error);
void PrintRequestError(int error);

int main(int argc, char* argv[])
{
    ModbusChannelHandle channel;
```



```
DeviceHandle device;
int result;

// Включаем библиотеку
ModbusLibraryInit();

// Создаем канал на COM1 и baudрейт 115200
channel = CreateChannel(1, 115200);

// Проверяем получилось ли
if (!IS_HANDLE_OK(channel))
{
    printf("Can't create channel\n");
    return -1;
}

// Подключаемся к устройству на канале с заданным номером
// Эта функция автоматически определяет название устройства
// и подгружает требуемую библиотеку .movilib
device = ConnectToDevice(channel, 1);

// Проверяем получилось ли
if (!IS_HANDLE_OK(device))
{
    PrintConnectError(device);
    DestroyChannel(channel);
    return -2;
}

// Проверяем совместимость заголовка и устройства
if (CheckDeviceDefinition(device, SampleDeviceName,
SampleDeviceVersion, SampleDeviceFirmware) != CHECK_OK)
{
    printf("Incompatible device\n");
    DestroyChannel(channel);
    return -3;
}

// Зададим какое-нить значение
if (RequestResultSuccess != (result =
WriteDeviceParamAsInteger(device, PID_SAMPLE_DEVICE_PARAMETER, 10)))
{
    PrintRequestError(result);
    DestroyChannel(channel);
    return -4;
}

printf("Parameter changed\n");
DestroyChannel(channel);
return 0;
}

void PrintConnectError(int error)
```



```
{
    switch(error)
    {
    case NO_CHANNEL:
        printf("Channel not opened\n");
        break;
    case NO_DEVICE:
        printf("Device not found\n");
        break;
    case UNKNOWN_DEVICE:
        printf("Device found, but unknown\n");
        break;
    case NO_LIBRARY_FOR_DEVICE:
        printf("Can't find library for detected device\n");
        break;
    }
}

void PrintRequestError(int error)
{
    switch(error)
    {
    case RequestResultMapNotLoaded:
        printf("Can't find library for device\n");
        break;
    default:
        printf("Modbus API function code returned %d\n", error);
    }
}
```

2. Устройство библиотеки

Данная библиотека спроектирована для облегчения работы с устройствами производства Movicom, точнее с устройствами, позволяющими соединение по протоколу modbus. Мы даем описание всех элементов представления для настройки устройства по протоколу modbus. Однако, намного удобнее использовать подобную библиотеку, особенно, если вы не обладаете собственной реализацией modbus клиента.

Многие параметры устройства при передаче по modbus используют единицы, которые не являются понятными человеку, потому что используют различные коэффициенты для получения целочисленного значения для передачи. Данная библиотека позволяет избежать подстановки различных коэффициентов из описания устройства, делая это за пользователя.



2.1. Карта параметров устройства

Основой данной библиотеки является карта параметров устройства, она позволяет пользователю не знать коэффициентов перевода, не задумываться о границах значений, не знать по большому счету протокол связи с устройством и адрес параметра и способ его передачи по протоколу.

2.1.1. Название карты параметров

Название карты параметров составляется по названию устройства, поэтому по файлам карты можно легко определить к какому устройству они относятся.

Название:

Название_устройства.Версия.Версия_программы

Т.е. для устройства MoviCAR v2.01 firmware 0.4 карта будет называться:

MoviCAR.2.01.0. 40

2.1.2. Файлы карты параметров

Карта параметров состоит из двух файлов.

- Заголовка для разработчика. (.h) Содержит макросы всех параметров, а также макросы заголовка устройства, для проверки.
- Файла карты. (.movilib) Содержит бинарные данные карты, которые подгружаются библиотекой при обнаружении устройства.

Таким образом для работы с устройством MoviCAR v2.01 firmware 0.4 требуется наличие файлов:

- MoviCAR.2.01.0. 40.h
- MoviCAR.2.01.0. 40.movilib

2.2. Типы, используемые в библиотеке

- **ModbusChannelHandle** – тип для манипулирования с открытым каналом связи. Используется для поиска новых устройств на канале
- **ModbusDeviceID** – идентификатор устройства в протоколе связи
- **DeviceHandle** – тип для манипулирования с устройством, т.е. для доступа к параметрам устройства
- **RequestResult** – результат запроса на изменение параметра

2.2.1. Тип RequestResult (Таблица результатов запроса параметров)

Содержит код результата операции. Существуют следующие коды:

- RequestResultSuccess (0x00)

Операция выполнена успешно

- RequestResultMapNotLoaded (0xF0)

Отсутствует карта памяти для устройства

- RequestCanceled (0xFF)

Запрос был отменен пользователем

- Request_ILLEGAL_FUNCTION (0x01)

Функция не поддерживается устройством (устройство не поддерживает параметры заданного типа modbus)

- Request_ILLEGAL_DATA_ADDRESS (0x02)

В устройстве отсутствует параметр с заданным адресом (основная причина, библиотека более новой версии, чем устройство при неизменном заголовке устройства)

- Request_ILLEGAL_DATA_VALUE (0x03)

Данные не могут быть заданы (они выходят за границы допустимых)

- Request_SLAVE_DEVICE_FAILURE (0x04)

Внутренняя ошибка устройства (обращаться к разработчику)

- Request_SLAVE_DEVICE_BUSY (0x06)

Невозможно обработать запрос (обращаться к разработчику)

- Request_MEMORY_PARITY_ERROR (0x08)

Ошибка настройки связи. (устройство не поддерживается библиотекой)



2.2.2. Тип ModbusDeviceID

Целочисленное значение идентификатора устройства в используемом протоколе связи. Для определения какой номер имеет ваше устройство обратитесь к описанию устройства.

2.2.3. Типы ModbusChannelHandle и DeviceHandle

Эти типы существуют для доступа к объектам, создаваемым библиотекой. Для получения объекта этого типа необходимо вызвать соответствующую функцию библиотеки. Не рекомендуется задавать значения этих типов самостоятельно.

2.3. Функции библиотеки

2.3.1. Функция ModbusLibraryInit

Сигнатура функции:

```
int ModbusLibraryInit(void)
```

Возвращает версию библиотеки. Вызывается перед началом работы с библиотекой.

2.3.2. Функция CreateChannel

Сигнатура функции:

```
ModbusChannelHandle CreateChannel(int ComPortNumber,  
unsigned int baudrate)
```

Открывает новый канал связи. Используется ком-порт компьютера с номером **ComPortNumber** и скоростью связи **baudrate**.

Возвращает манипулятор для созданного канала связи либо **WRONG_HANDLE(-1)**, если не удалось открыть канал связи.

Рекомендуется пользоваться макросом **IS_HANDLE_OK** для проверки успешности выполнения функции.

2.3.3. Функция DestroyChannel

Сигнатура функции:



```
void DestroyChannel (ModbusChannelHandle channel)
```

Закрывает открытый канал связи. Параметры: манипулятор открытого канала связи.

2.3.4. Функция ConnectToDevice

Сигнатура функции:

```
DeviceHandle ConnectToDevice (ModbusChannelHandle  
channel, ModbusDeviceID devID)
```

Функция создает соединение с устройством **devID** на канале **channel**.

Функция определяет наличие устройства на канале связи, определяет название устройства и находит библиотеку параметров для этого устройства.

Возвращает манипулятор для работы с устройством, либо код ошибки. Для проверки, является ли возвращенное число корректным манипулятором, следует использовать макрос **IS_HANDLE_OK**.

Может вернуть один из следующих кодов ошибок:

- **NO_CHANNEL(-1)** отсутствует либо был закрыт канал связи с устройством
- **NO_DEVICE(-2)** нет устройства с заданным номером
- **UNKNOWN_DEVICE(-3)** устройство не вернуло свое имя, т.е. не может работать с данной библиотекой
- **NO_LIBRARY_FOR_DEVICE(-4)** отсутствует карта параметров устройства

2.3.5. Функция CheckDeviceDefinition

Сигнатура функции:

```
int CheckDeviceDefinition (DeviceHandle device, const  
char *name, const char *version, const char *firmware)
```

Функция проверяет, является ли заголовочный файл, который вы используете в программе, файлом для устройства, которое было найдено на канале. Это необходимо, если у вас есть большое количество карт параметров для различных устройств.

Возвращает:

- **CHECK_OK(0)** – проверка прошла успешно



- CHECK_FAILED(-1) – устройство и заголовок не совместимы

2.3.6. Функция SetDeviceTimeout

Сигнатура функции:

```
void SetDeviceTimeout (DeviceHandle device, int  
timeoutMs)
```

Функция задает таймаут для функций работы с параметрами. Таймаут должен зависеть от скорости канала связи и устройства. Можно использовать значение по умолчанию, либо задавать свое. Подбирается эмпирически. Стоит заметить, что обычно задержки в шине USB компьютера составляют порядка 1-2мс. Т.е. при мгновенной реакции устройства и быстром канале связи, при использовании таких устройств, все равно таймаут нельзя ставить меньше 4мс.

Ничего не возвращает.

2.3.7. Функции ReadDeviceParamAsXXX

Сигнатура функции:

```
RequestResult ReadDeviceParamAsInteger (DeviceHandle  
device, int pid, int &parameter)
```

```
RequestResult ReadDeviceParamAsDouble (DeviceHandle  
device, int pid, double &parameter)
```

Функции читают параметр с заданным номером. Номера параметров задаются макросами в файле-заголовке устройства. Описание свойств с указанием имен макросов можно найти в описании устройства.

Описание возвращаемых значений можно найти в разделе **Тип RequestResult (Таблица результатов запроса параметров)**.

2.3.8. Функции WriteDeviceParamAsXXX

Сигнатура функции:



```
RequestResult WriteDeviceParamAsInteger (DeviceHandle  
device, int pid, int parameter)
```

```
RequestResult WriteDeviceParamAsDouble (DeviceHandle  
device, int pid, int parameter)
```

Все аналогично функциям чтения. Записывают значения параметров. Если задается значение, выходящее за рамки допустимых границ, то значение автоматически обрезается до границы.