



**Управление роботами с
помощью радиомодулей
R-Cat производства
Мовиком**

Мовиком, 2006-2007
Версия 1.4
04.07.2007



Оглавление

Вводные замечания	3
Библиотека MoviRadio Universal Multi 6.0	3
Инициализация и минимальный код для работы с USB радио модулями производства Мовиком.....	3
Функции библиотеки MoviRadio UM	4
Инициализация	4
Функция movi_radio_create_channel.....	4
Функция movi_radio_create_serial_channel.....	4
Функция movi_radio_is_channel_created	5
Функция movi_radio_destroy_channel.....	5
Функция movi_radio_shutdown	5
Поиск устройств.....	5
Класс CMoviRadioDevice	5
Функция movi_radio_enumerate.....	6
Функция movi_radio_next_device	6
Пример использования.....	6
Посылка и прием сообщений	7
Функция movi_radio_send_message.....	7
Функция movi_radio_get_count	7
Функция movi_radio_get_message	8
Функция movi_radio_clear.....	8
Функция movi_radio_clear_all.....	8
Пример приема сообщений	9
Протокол передачи.	9
Инициализация канала	9
Формат сообщений	10
Пример составленного сообщения.....	10
Прием сообщений.....	10
Логика работы или как работать с радиомодулями без библиотек.....	11
Работа с компонентами Movicom, поддерживающими стандартный протокол.....	12
Сообщения, используемые компонентами	12
Задание скоростей роботу.....	12
Alive сообщение, получение информации.....	12



Вводные замечания

Данный документ детально описывает возможности программного общения с роботом помощью библиотеки MoviRadioUM.dll, версия 6.2.0.1, и без ее использования.

Библиотека MoviRadio Universal Multi 6.0

Инициализация и минимальный код для работы с USB радио модулями производства Мовиком.

Note: Перед началом работы, необходимо установить на компьютер драйверы устройства.

Интерфейс библиотеки связи с роботами описан в заголовочном файле MoviRadio.h.

При разработке в среде Microsoft Visual Studio C++, достаточно добавить этот заголовочный файл и поместить файлы MoviRadio.lib и MoviRadio.dll в папку проекта.

В начале файла добавляем

```
#include "MoviRadioUM.h"
```

В тексте программы:

```
int iDevNum = 0;  
  
// Подключаемся к первому найденному устройству  
int iChHandle = movi_radio_create_channel(iDevNum);  
  
if (movi_radio_is_channel_created(iChHandle)) {  
    // Канал успешно создан  
}  
else {  
    // Неудачно (устройство не обнаружено или занято)  
}
```

Перед завершением программы:

```
movi_radio_destroy_channel(iChHandle);
```



Функции библиотеки MoviRadio UM

Инициализация

Функция `movi_radio_create_channel`

Функция инициализирует радиомодуль производства Movicom.

```
int movi_radio_create_channel (int iDevNum);
```

Параметры: `iDevNum` - номер устройства в списке нумератора

Возвращает: Отрицательное значение - ошибка создания канала. Положительное значение - номер открытого канала.

При инициализации с помощью этой функции запускается внутренняя система приема сообщений, которая независимо от остальной программы разбирает принятую информацию на отдельные сообщения и складывает все принятые полные сообщения в буфер (подробнее о чтении сообщений в разделе Посылка и прием сообщений).

Функция `movi_radio_create_serial_channel`

Функция инициализирует устройство стандартный последовательный порт. Настройки порта baudrate: 115200, data bits: 8, stop bits: 1, parity: none, handshake: none.

```
int movi_radio_create_serial_channel (int iComPortNum);
```

Параметры: `iComPortNum` - номер последовательного порта

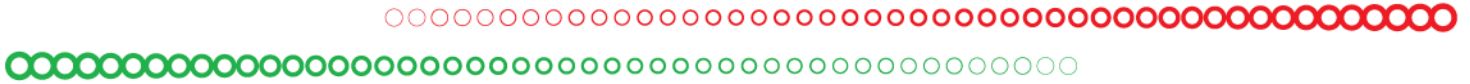
Возвращает: Отрицательное значение - ошибка создания канала. Положительное значение - номер открытого канала

Функция действует аналогично функции `movi_radio_create_channel`.

```
int iComNum = 1;

// Подключаемся к COM1
int iChHandle = movi_radio_create_serial_channel (iComNum);

if (movi_radio_is_channel_created(iChHandle)) {
    // Канал успешно создан
}
else {
    // Неудачно (устройство не обнаружено или занято)
}
```



Функция `movi_radio_is_channel_created`

Проверяет, был ли открыт канал связи.

```
bool movi_radio_is_channel_created (int iChNum);
```

Параметры: `iChNum` – Номер канала, возвращенный функцией `movi_radio_create_channel` или функцией `movi_radio_create_serial_channel`.

Возвращает: Признак успешно созданного канала

Функция `movi_radio_destroy_channel`

Закрывает канал связи.

```
void movi_radio_destroy_channel(int iChNum);
```

Параметры: `iChNum` – Номер канала.

Функция `movi_radio_shutdown`

Закрывает все открытые каналы связи

```
void movi_radio_shutdown ();
```

Параметры: нет

Поиск устройств

Радиомодули USB Movicom 433MHz и 2.4GHz допускают автоматическое нахождение и нумерацию. В процессе нумерации возможно определение параметров устройств.

Класс `CMoviRadioDevice`

Хранит информацию о найденном устройстве

Данные:

```
int iDevNum;
```

Номер устройства, полученный нумератором.

```
bool bDevFree;
```

Признак устройства свободного для работы (смотри замечание ниже).

```
char SerialNumber[16];
```

Серийный номер устройства, 16 байт означающие уникальный идентификатор устройства. Если устройство инициализировано этой или другой программой, то серийный номер будет состоять из нулей.



```
char Description[64];
```

Текстовая строка описания устройства.

Функция `movi_radio_enumerate`

Начало процесса поиска устройств

```
int movi_radio_enumerate();
```

Возвращает: Количество устройств.

Функция `movi_radio_next_device`

Функция заполняет структуру `CMoviRadioDevice` информацией о следующем найденном устройстве

```
bool movi_radio_next_device(CMoviRadioDevice * pNextDev);
```

Параметры: `pNextDev` – указатель на структуру типа `CMoviRadioDevice`, в которую будет помещена информация о найденном устройстве

Возвращает: Признак того, была ли записана информация о новом устройстве (т.е. `false` в случае если уже получена информация о всех найденных устройствах)

Пример использования

Следующий код получает информацию о всех подключенных устройствах

```
// Произведем поиск устройств, найдем их количество
int iDevCount = movi_radio_enumerate();

// Для всех найденных устройств узнаем информацию об них
for (int i=0; i < iDevCount; i++)
{
    // Здесь хранится информация об устройствах
    CMoviRadioDevice mrDevices[100];

    if (movi_radio_next_device( mrDevices + i ))
    {
        // Выводим информацию об очередном найденном устройстве
        std::cout << mrDevices[i].iDevNum << "." << \
mrDevices[i].Description << std::endl;
    }
}
}
```



Посылка и прием сообщений

Библиотека обеспечивает простой интерфейс для обмена сообщениями через канал связи.

Функция `movi_radio_send_message`

Посылает сообщение установленного формата через канал связи.

```
void movi_radio_send_message( int iChNum,  
                             char robot_number,  
                             char message_number,  
                             void *msg);
```

Параметры: `iChNum` - номер канала связи, полученный при создании канала.

`robot_number` - номер робота, которому будет послано сообщение (0 - `message_max_number`),

`message_number` - номер посылаемого сообщения (0 - `robot_max_id`),

`msg` – тело сообщения, указатель должен указывать на буфер размера `std_message_size` байт

Пример использования:

```
// Тело сообщения  
char buffer[std_message_size];  
  
// Некоторое посылаемое значение  
int data_to_send = 100;  
  
// Записываем его в буфер  
buffer[0] = data_to_send & 0xFF;  
buffer[1] = (data_to_send >> 8) & 0xFF;  
buffer[2] = (data_to_send >> 16) & 0xFF;  
buffer[3] = (data_to_send >> 24) & 0xFF;  
  
// Посылаем сообщение  
movi_radio_send_message(iChNum, 1, 0, buffer);
```

Функция `movi_radio_get_count`

Возвращает количество принятых сообщений в буфере.

```
int movi_radio_get_count(int iChNum);
```



Функция `movi_radio_get_message`

Читает из буфера (очереди) первое сообщение, возвращает количество сообщений до прочтения. Т.е. нулевое значение означает, что сообщение не было считано.

```
int movi_radio_get_message(int iChNum,  
                           char *robot_number,  
                           char *message_number,  
                           char *msg);
```

Параметры: `iChNum` - номер канала связи, полученный при создании канала.

`robot_number` – указатель, куда будет записан номер робота, от которого получено сообщение (0 - `message_max_number`),

`message_number` – указатель, куда будет записан номер принятого сообщения (0 - `robot_max_id`),

`msg` – тело сообщения, указатель должен указывать на буфер размера `std_message_size` байт

Функция `movi_radio_clear`

Очищает очередь принятых сообщений канала `iChNum`.

```
void movi_radio_clear(int iChNum);
```

Функция `movi_radio_clear_all`

Очищает очередь принятых сообщений всех каналов

```
void movi_radio_clear_all();
```



Пример приема сообщений

Следующий код читает все сообщения из буфера и запускает некоторую функцию обработки.

```
// Данные по сообщениям
char rnum;
char mnum;
char buffer[std_message_size];

// Сначала проверим наличие сообщений
if (movi_radio_get_count(iUsedCh) return ;

while (movi_radio_get_message(iUsedCh, &rnum, &mnum, buffer)
{
    // Прочитано сообщение запускается некий обработчик
    if (!OnMessageFunction(rnum, mnum, buffer))
    {
        // В случае, если надо остановиться, останавливаемся очищая
        // необработанные сообщения
        movi_radio_clear(iUsedCh);
        break;
    }
}
```

Протокол передачи.

Библиотека MoviRadioUM использует стандартный протокол передачи информации, этот протокол поддерживается, например роботами футболистами Movicom и программой управления роботами от джойстиков.

Для возможности работы с такими устройствами и программами ниже приводится описание протокола.

Инициализация канала

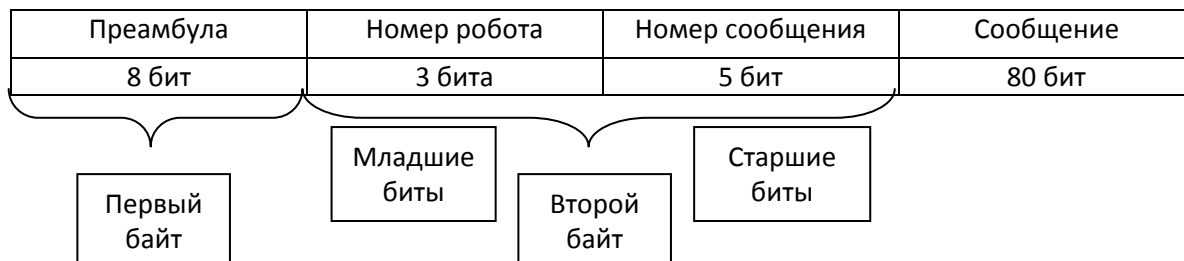
Радиомодули Movicom (USB, UART), версия для роботов используют следующие настройки инициализации COM порта/UART:

Baud rate 115200
Data bits 8
Stop bits 1
Parity None
Handshake Off



Формат сообщений

Посылка сообщений по радио от компьютера к удаленному устройству осуществляется в установленном формате.



Обобщенным id сообщения можно считать второй байт сообщения, разделенный на номер сообщения и номер робота.

Преамбула для сообщений от компьютера к роботу равна 0xDB

Преамбула для сообщений от робота к компьютеру равна 0xDA

Пример составленного сообщения

Чтобы послать сообщение вида:

Номер робота: 1

Номер сообщения: 6

Байты сообщения: 0x0A 0x70 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00

В порт надо записать: 0xDB 0x31 0x0A 0x70 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Прием сообщений

Сообщения от робота отправляются с преамбулой 0xDA, поэтому, пользователю необходимо в потоке информации с COM-порта искать байт преамбулы и отсчитывать от него 11 байт для получения сообщения.



Логика работы или как работать с радиомодулями без библиотек

Радиомодули имеют логический интерфейс COM-порта (UART'a для малых модулей). Для работы непосредственно с этими логическими интерфейсами требуется знать несколько простых правил.

1. Все последовательные интерфейсы радиомодулей имеют фиксированные настройки передачи, а именно
 - Baud rate 115200
 - Data bits 8
 - Stop bits 1
 - Parity None
 - Handshake Off
2. Радиомодуль не будет передавать сообщение пока не найдет в потоке идущей к нему информации байт преамбулы и все байты сообщения (т.е. нужное количество последовательно идущих байт, несовпадающих с байтом преамбулы)
3. Байты преамбулы:
 - a. USB радио: 0xDB для послылки (пользователь должен записать в последовательный порт), 0xDA для приема (т.е. маркирует им принятые сообщения, и пользователь их может прочитать из последовательного порта)
 - b. OEM (UART) радио: 0xDA для приема, 0xDB для послылки
 - c. Байты преамбулы – это управляющие команды для радиомодуля. Они не передаются по радиоканалу. Т.е. если в UART одного OEM радиомодуля было записано 0xDA и затем сообщение, то в другом OEM радиомодуле пользователь прочитает 0xDB и посланное сообщение.
4. Адресация.
 - a. При работе непосредственно с последовательным портом не используется адресация. Посланное сообщение будет записано и выдано пользователям всех радиомодулей в зоне приема, не зависимо от их типа (USB или OEM).



Работа с компонентами Movicom, поддерживающими стандартный протокол

Сообщения, используемые компонентами

Сообщения, используемые в управлении роботами с помощью программы joystick_m

Задание скоростей роботу

Номер сообщения 0

Для управления футболистом сообщение должно содержать значения скоростей в мм/с.

Тело сообщения:

Скорость левого колеса (16 бит)	Скорость правого колеса (16 бит)
---------------------------------	----------------------------------

Младший байт скорости должен идти первым, старший – вторым.

Alive сообщение, получение информации

На это сообщение робот должен отвечать, передавая любую информацию о себе в первом байте тела сообщения (программа joystick_m может отображать это значение).

Номер сообщения 0x1F

Тело сообщения:

- от компьютера к роботу – произвольное,
- от робота к компьютеру: первый байт – отображаемая информация.

Робот должен по приему этого сообщения сформировать ответ и отослать обратно с таким же номером.