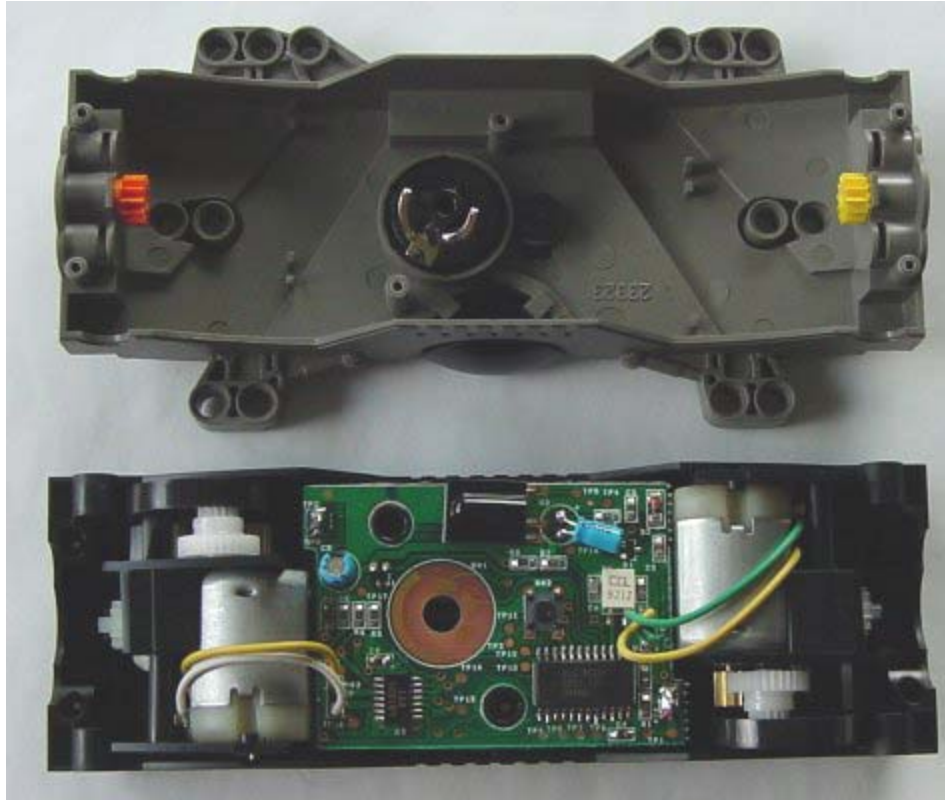


RCX Control of Bionicle 8539 Manas Receivers

The 8539 Manas set comes with 2 receiver units and 2 transmitter units. The receivers feature two drive axel sockets on each end, a 'through axle' hole and an 'end axle' hole. The axis of the through axles and end axles are 90° to each other. The two are geared differently so you can choose higher speed or higher torque. **The gear ratio between the two types of axle holes is about 5:2. (The 'through axle' turns faster), on a light load, the speed is roughly 300/120 RPM.¹**



The transmitter offers channels 1, 2 or 3 or 'All'. A rotary switch must be set on both the transmitter and the receiver to get correct operation.

Each channel is differentiated by both an address in the message and by the rate at which messages are sent. The messages are very short in comparison to the time between messages which helps minimize the chances of a 'collision'. The receivers have error checking and

simply ignore bad messages. Each button press on the transmitter sends a continuous stream of messages. When the last button is released, a set of three 'no buttons pressed' messages are transmitted before the transmitter shuts down. If the 'no buttons pressed' message is missed by the receiver, it times out after about half a second and turns the motors off anyhow.

The transmitters transmit short messages of two bytes encoded as 8 bit + odd parity. The transmissions use 76kHz modulated IR at a baud rate of 4800, 25% duty cycle. The messages, which consist of two bytes are best thought of as four 4 bit nibbles. byte 1 - high nibble : byte 1 - low nibble : byte 2 - high nibble : byte 2 low nibble. These four nibbles are used as follows;

address : orange : yellow : check

The address nibble can be;

- 4 - all
- 5 - Ch. 1
- 6 - Ch. 2
- 7 - Ch. 3

The orange and yellow nibbles reflect the current button state:

- 0 - no button pressed
- 7 - forward button

RCX Control of Bionicle 8539 Manas Receivers

8 – immediate off ⁱⁱ

F - reverse button

The check nibble contains a check digit:

Checkdigit = 0 - (address + yellow + orange)

Thus, for example:

50 74 is forward yellow button on channel 1
6F 0B is reverse orange button on channel 2
40 0C no buttons pressed on 'All' channel

The message rates are as follows:

Ch. 1 - 19.6 Hz
Ch. 2 - 14.8 Hz
Ch. 3 - 11.6 Hz
'All' - 19.6 Hz

The following NQC program controls six Manas motors (three receivers):

```
#define MANAS_1 5
#define MANAS_2 6
#define MANAS_3 7
#define EM_FLOAT 0
#define EM_OFF 8
#define EM_FWD 7
#define EM_REV 15

int em1,em2,em3,em4,em5,em6; //Global motor control locations

task main()
{
// This demo starts the manas comms task, then loops forever.
// In the loop, it steps through three control phases setting
// the motors to off, and then paired forward and reverse.

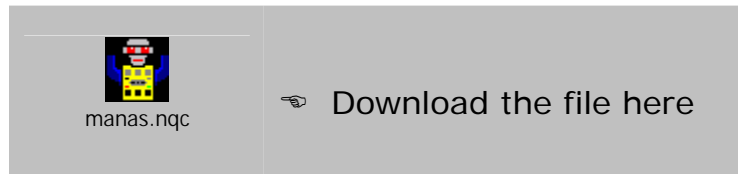
start manas;
while (true)
{
em1=EM_OFF;
em2=EM_OFF;
em3=EM_OFF;
em4=EM_OFF;
em5=EM_OFF;
em6=EM_OFF;

Wait(100);

em1=EM_FWD;
em2=EM_REV;
em3=EM_FWD;
em4=EM_REV;
em5=EM_FWD;
em6=EM_REV;

Wait(100);

em1=EM_REV;
em2=EM_FWD;
em3=EM_REV;
```



RCX Control of Bionicle 8539 Manas Receivers

```
    em4=EM_FWD;
    em5=EM_REV;
    em6=EM_FWD;

    Wait(100);
}
}

// The manas task continuously resends the motor settings.
// When the RCX is halted, this task stops and the manas
// motors automatically shut off

task manas()
{
    while (true)
    {
        // First ensure serial settings are still ok
        SetSerialComm(SERIAL_COMM_4800|SERIAL_COMM_DUTY25|SERIAL_COMM_76KHZ);
        SetSerialPacket(SERIAL_PACKET_DEFAULT);

        // Set to low power so we don't run the battery down too quick
        SetTxPower(TX_POWER_LO);

        // Set the first unit two message bytes and send them
        SetSerialData(0,MANAS_1*0x10+em1);
        SetSerialData(1,em2*0x10+0x10-((MANAS_1+em1+em2)&0xf));
        SendSerial(0,2);

        // Set the second unit two message bytes and send them
        SetSerialData(0,MANAS_2*0x10+em3);
        SetSerialData(1,em4*0x10+0x10-((MANAS_2+em3+em4)&0xf));
        SendSerial(0,2);

        // Set the third unit two message bytes and send them
        SetSerialData(0,MANAS_3*0x10+em5);
        SetSerialData(1,em6*0x10+0x10-((MANAS_3+em5+em6)&0xf));
        SendSerial(0,2);

        // Delay for a while so we can resend them regularly
        Wait(10);
    }
}
```

John Barnes

<http://news.lugnet.com/robotics/?n=15809>

ⁱ [Hao-yang Wang](#)

<http://news.lugnet.com/robotics/?n=15820>

ⁱⁱ [Dennis Williamson](#)

<http://news.lugnet.com/robotics/?n=16045>