MIKROKOPTER SERIAL CONTROL TUTORIAL

CREATED BY

JOHN C. MACDONALD

Ira A. Fulton College of Engineering and Technology

Department of Electrical and Computer Engineering

The accompanying files for this tutorial can be found at http://hdl.lib.byu.edu/1877/2748.

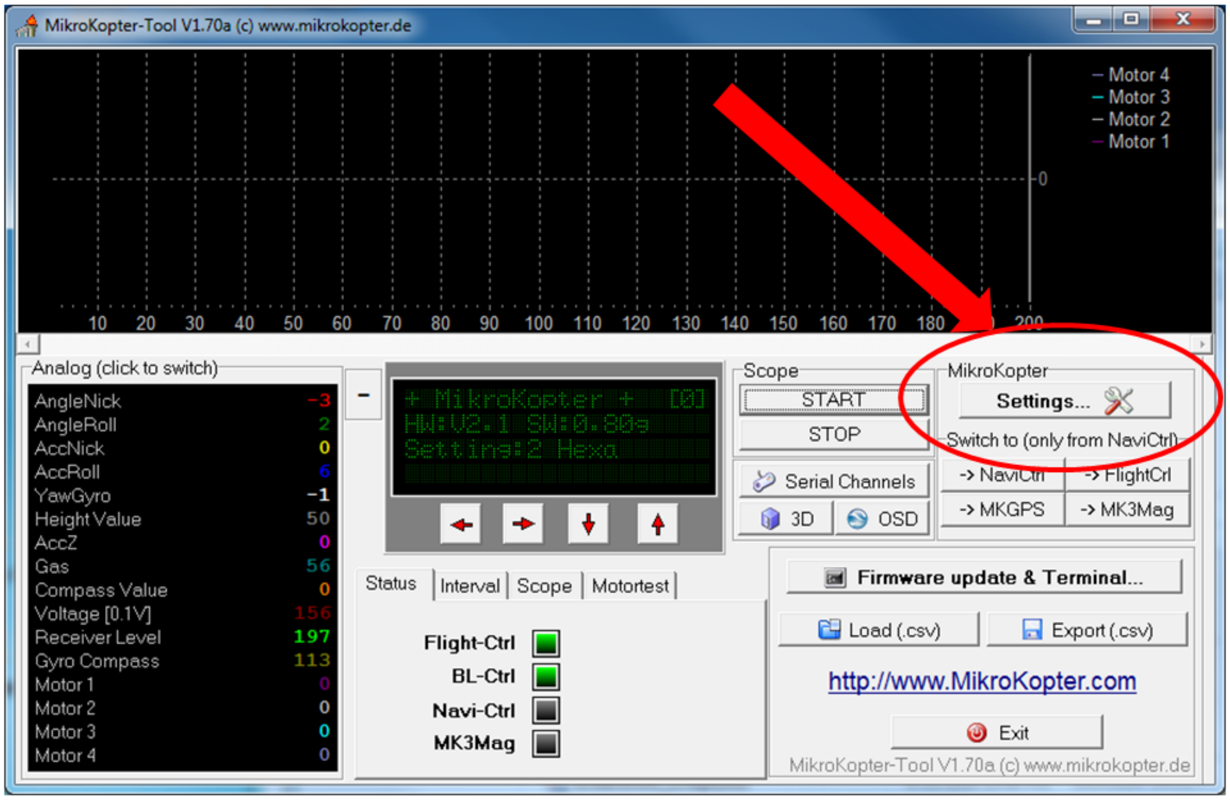
How to Control the MK HexaKopter with the Serial Interface

This tutorial is intended to introduce someone very new to the MK HexaKopter (MK or Hexa, for short) to the important aspects of controlling the MK with the serial interface to the FlightControl (FC) board. This tutorial was developed using the following hardware/software:

- FlightControl ME V2.0
- MKUSB
- FC firmware V0.80g
- HexacopterController.sln (A MS Visual Studio project written in C# that should be found somewhere near this document...)

To become familiar with how to control the Hexa via serial, do the following:

1. Take a look at the webpage http://www.mikrokopter.de/ucwiki/en/SerialProtocol. You'll find there some very useful tables explaining various serial commands understood by the MK. We're particularly interested in the ExternControl command used to pass the ExternControl Struct. For example implementations, see http://svn.mikrokopter.de/filedetails.php?repname=Projects&path=%2FMoteCtrl%2FSources%2Fwiimote.h (source code for the MoteCtrl project) and http://github.com/ligi/Riddim/blob/master/fc.h (source code for the Riddim project). This struct has the following members, 11 bytes in all:
    a. Struct members:
        i. unsigned char Digital[2] //2 bytes for something... don't know what so just set to zero
        ii. unsigned char RemoteTasten //Another mystery byte... set to zero
        iii. signed char Nick //Nick = Pitch
        iv. signed char Roll //Roll = Roll :-)
        v. signed char Gier //German for yaw...
        vi. unsigned char Gas //i.e. Throttle
        vii. signed char Higt //Apparently resets the baro sensor... for indoor flight set to 0
        viii. unsigned char Free //mystery byte... set to zero
        ix. unsigned char Frame //We're not using this; sent from FC to basestation to confirm reception
        x. unsigned char Config //Needs to be set to 1 before sending to FC
    b. In the HexacopterControl project associated with this tutorial, ExternControl is implemented as a class in ExternControl.cs.
2. In order to send serial commands to the flight controller, a parameter in the FC software must be set to a value greater than 128.The MikroKopter-Tool software allows you to tie this parameter to a switch on you RC transmitter. To set this up:
    a. Connect your MK to the Tool software using the MKUSB (converts USB to serial).
    b. Open the "Settings…" menu as highlighted in Figure 1 below:



```
+ MikroKopter +  [0]
HW:V2.1 SW:0.80a
Setting:2 Hexa
```

AngleNick	-3
AngleRoll	2
AccNick	0
AccRoll	6
YawGyro	-1
Height Value	50
AccZ	0
Gas	56
Compass Value	0
Voltage [0.1V]	156
Receiver Level	197
Gyro Compass	113
Motor 1	0
Motor 2	0
Motor 3	0
Motor 4	0

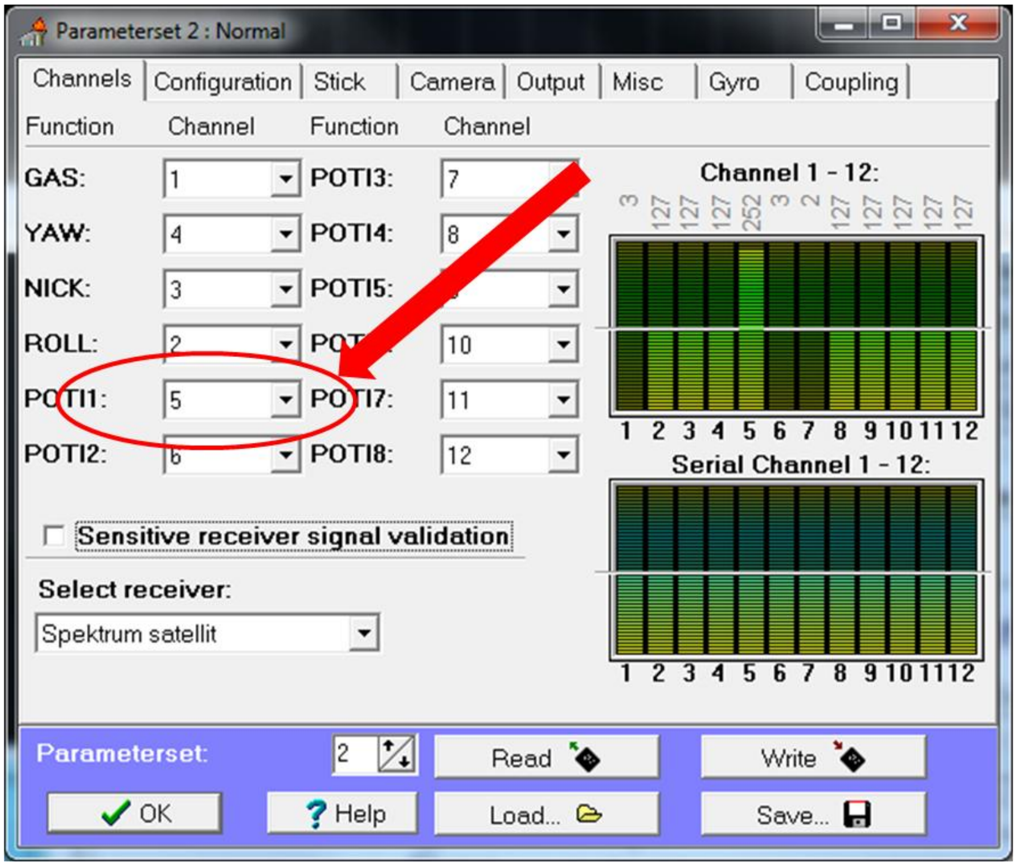
Flight-Ctrl	🟩
BL-Ctrl	🟩
Navi-Ctrl	■
MK3Mag	■

Figure 1: The "Settings…" Menu

c. Select the "Channels" tab. This should be a familiar menu already if you've configured your RC transmitter. Identify a channel you'd like to use for the parameter that needs to be set high. Specifically, the parameter needs to be set to be greater than 128. One of the two-position switches on the RC Tx should work fine for this. In Figure 2 below we have the two position switch associated with channel 5 of our Tx set to high. Notice (as highlighted in Figure 2 below) that the software associates "Poti1" with this channel.

d. Now select the "Stick" tab. In the drop down menu next to "External Control:" select the "Poti" associated with the channel you identified in the previous step. Finally, don't forget to click "Write" to save this setting to your MK (see Figure 3 below).

3. With your RC Tx configured, you're ready to try sending serial commands to the Hexa. For our first tests we put a metal bar through the landing gear of the MK with something heavy on the ends of the bar (undergrad, intern, etc...) to hold the MK in place. To send your first commands via serial, do the following:
   a. Power up the MK and the RC Tx. (Calibrate the gyros, etc. as you would do before any flight)
   b. Establish a serial connection between your computer and the MK. (In our experiments = Plug the USB cable into your computer that is connected to the MKUSB which is plugged into the 10-pin serial connection on the FC.)
   c. Open up the Windows "Device Manager" or something like that to identify which COM Port your computer has assigned to the MK connection.
   d. Open and run the HexacopterControl project. (Some comments on the project contents will follow, but I figured you'd like to see something interesting happen first...)
      i. In the GUI that pops up (See Figure 4 below) you will see several fields in the lower left containing COM Port configuration info. These should be the settings expected by the MK. The only one to change should be the COM Port Name your computer assigned to the MKUSB link.



Parameterset 2 : Normal

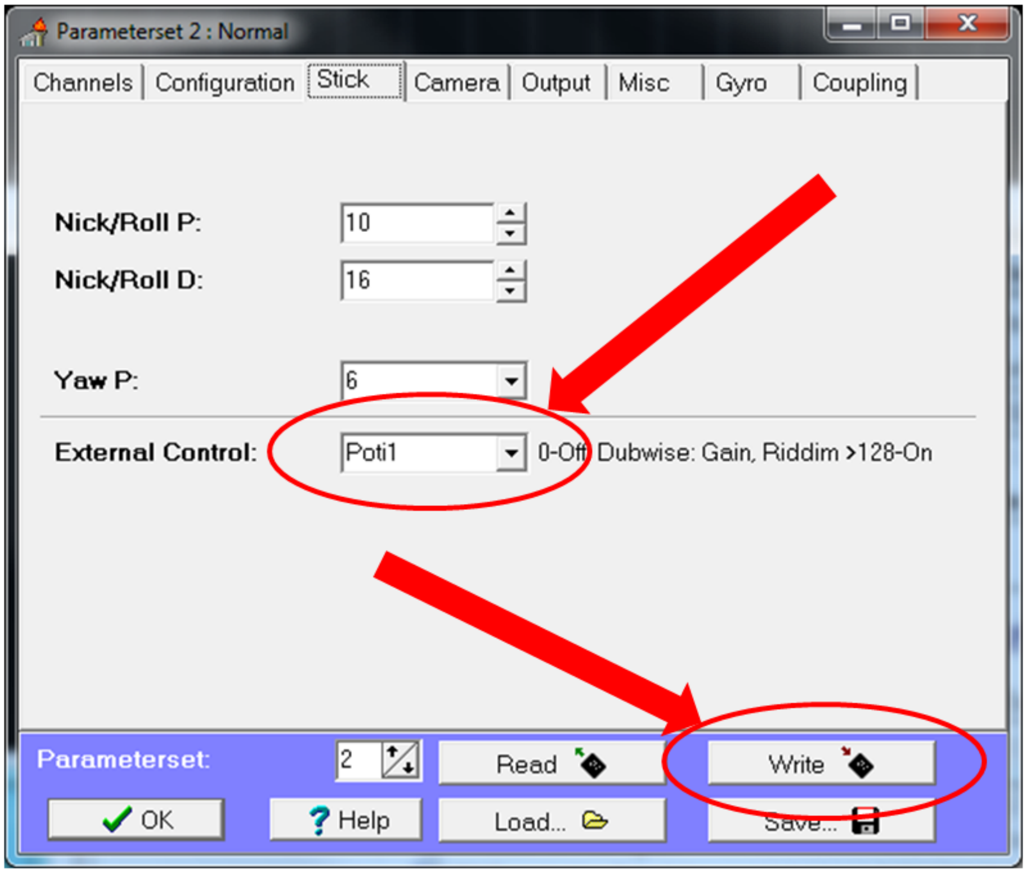
Channels | Configuration | Stick | Camera | Output | Misc | Gyro | Coupling

Function	Channel	Function	Channel
GAS:	1	POTI3:	7
YAW:	4	POTI4:	8
NICK:	3	POTI5:	
ROLL:	2	POT	10
POTI1:	5	POTI7:	11
POTI2:	6	POTI8:	12

☐ Sensitive receiver signal validation

Select receiver: Spektrum satellit

Figure 2: “Channels” tab with desired channel for ExternControl parameter set high



Parameterset 2 : Normal

Channels | Configuration | Stick | Camera | Output | Misc | Gyro | Coupling

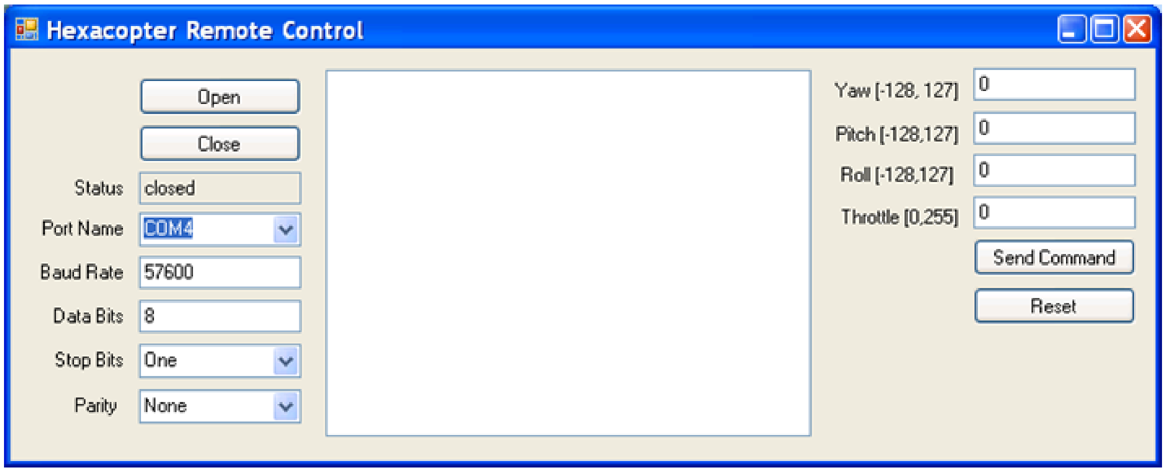
Nick/Roll P: 10

Nick/Roll D: 16

Yaw P: 6

External Control: Poti1 0-Off Dubwise: Gain, Riddim >128-On

Figure 3: “Stick” tab with highlighted areas as discussed above


Hexacopter Remote Control

Status	closed
Port Name	COM4
Baud Rate	57600
Data Bits	8
Stop Bits	One
Parity	None

Yaw [-128, 127]	0
Pitch [-128,127]	0
Roll [-128,127]	0
Throttle [0,255]	0

Figure 4: The HexacopterControl Project GUI

ii. Above those COM Port properties are buttons for opening and closing communication with the COM Port. Go ahead and click "Open."

iii. In the center of the GUI you should start to see the data that the MK is apparently streaming out of the serial port. Judging by the tables on that serial protocol webpage, this seems to be some data intended for the MK3MAG board (which we don't have...). Just ignore this stuff for now.

iv. Start your motors spinning from the RC Tx. Flip the switch you configured to enable ExternControl commands. Leave the throttle at its minimum on the RC Tx.

v. On the right side of the GUI there are some fields where you can enter commands for yaw, pitch, roll, and throttle. For starters, enter "127" in the "pitch" field and click the "Send Command" button (DON'T FORGET TO HAVE SOMETHING HOLDING THE MK IN PLACE).

vi. You should see and hear the front three motors of the Hexa slow down and the back three speed up. Assuming you do nothing else, the MK should remain in this state for about 5 seconds and then it will return to all motors equal and minimum throttle. This is because the FC firmware is set up to execute the serial commands for roll/pitch/yaw/throttle for 5 seconds unless it receives new serial commands. After those 5 seconds the MK reverts to control from the RC Tx.

vii. Enter the same pitch command again, but this time shortly after clicking the button to send the command over serial, also pull all the way back on the pitch stick on the RC Tx. You will hear the motor speeds change as before, but when you pull back on the stick you will hear the motors go back to all equal. This is because the FC firmware is set up to add roll/pitch/yaw commands sent over serial to whatever it receives over RC. In other words, the RC Tx is always in the loop (though the best you can do is fight the serial commands, unless...)

viii. Enter the same pitch command again, but this time shortly after clicking the button to send the command over serial, deactivate the switch on the RC that enables ExternControl commands. You should hear the motor speeds change to implement the pitch command until you flip the switch, at which point you will hear the motors return to all equal at min throttle. If you flip the ExternControl switch on the RC Tx back on before the 5 seconds are up, you'll also hear the motors pitching again. This demonstrates the effect of that switch.

ix. You can keep experimenting if you want, but this demonstrates the basic ideas. ONE NOTABLE EXCEPTION: The FC treats the thrust commands from serial differently than serial commands for roll/pitch/yaw. Instead of adding the serial and RC throttle commands it takes the minimum of the two. This lets you use the RC throttle stick to set a maximum throttle allowed for serial control. However, if you're not careful it creates a potential hazard. If you have the RC throttle nice and high and are sending serial commands and those serial commands are interrupted or the 5 seconds otherwise expire, your MK might suddenly try to shoot through the roof (assuming you're indoors).

4. Now, to get down to the nuts and bolts we should discuss some of the source code used in the HexacopterControl project...
   a. ExternControl.cs.
      i. Nothing too exciting here – just some constructors and some getters and setters. Just important to understand that this is the 11 byte chunk of data the MK expects from you if you want to control it via serial.
      ii. Note that in order for an extern control command to be accepted by the FC, the "Config" byte must be set to a non-zero value.
   b. FlightControllerMessage.cs. A few important things to note here:
      i. First look at the CreateMessage function on line 16. Compare lines 21-24 with the tables back on http://www.mikrokopter.de/ucwiki/en/SerialProtocol. We're formatting the first part of the serial command here. Then we're tacking on the data in line 27 followed by the two checksum bytes and a carriage return (lines 30-31). The data needs to be encoded before sending and there's also a little black magic to create the checksum bytes...
      ii. To encode the data, see the Encode64 function at line 54. This is a magical black box described in the serial protocol wiki-page.
      iii. The checksum bytes are calculated with the getCRCBytes function that begins at line 110. This function is also described on the serial protocol wiki-page. One important thing to note is back on line 30; all the preceding bytes in the serial command (header material + data) are sent as input to this function.
      iv. For completeness, the corresponding functions for receiving (line 36) and decoding (line 78) messages sent over serial are included in this file.
   c. Other files (ConsoleWriter.cs, Program.cs, etc...) just contain stuff that is specific to this little test program. The important parts are in the preceding two files.