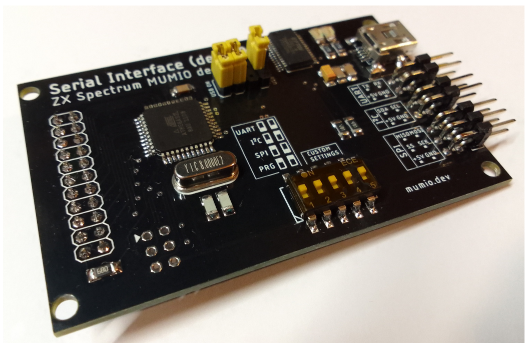
mumio

Serial Interface

DEVICE #03



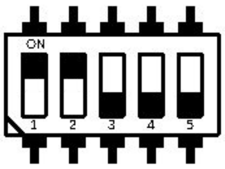
UART, I²C and USB serial interface. Allows to easy send/receive data using IN and OUT BASIC commands.

Device requires to be mounted on I/O Port (MUMIO dev. #00)!

First steps

Communicate with device

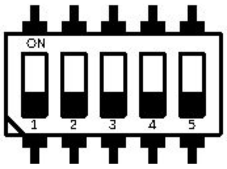
1. Using switches on I/O Port (dev. #00), set port address to **163** (binary: 10100011).
2. Mount device on the I/O port board, insert them into the edge connector of the computer, and turn on the power.
3. Set microswitch to PRG mode (bin: 11000), as on picture below:



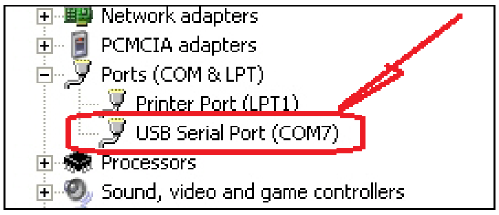
4. Enter and run terminal BASIC program, listed in "PRG mode" chapter below.
5. Enter "ver" command and press ENTER. Device shows you its version number.

Communicate ZX Spectrum to PC computer via USB interface

1. Set microswitch to UART mode (bin: 00000), as on picture below:

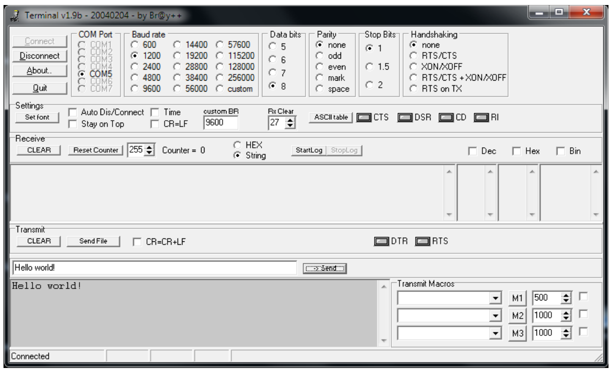


2. Connect device via USB cable to PC computer. In the Device Manager, check the COM port number under which the device was installed in the system (e.g. COM7).
3. Install and run Br@y++ Terminal (https://sites.google.com/site/terminalbpp/).



Network adapters
PCMCIA adapters
Ports (COM & LPT)
Printer Port (LPT1)
USB Serial Port (COM7)
Processors
Sound, video and game controllers

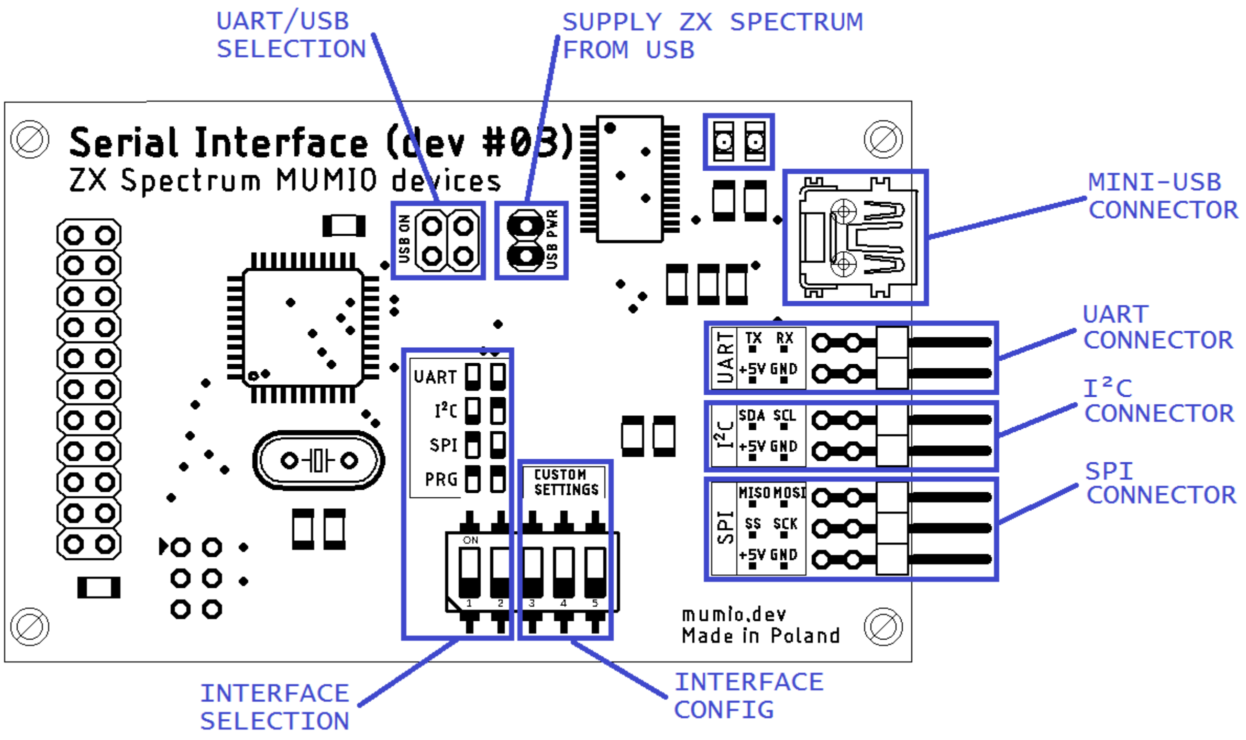
4. Install and run Br@y++ Terminal (https://sites.google.com/site/terminalbpp/). Set COM parameters to 1200,8,none,1 and press **Connect** button.



COM Port: COM5
Baud rate: 1200
Data bits: 8
Parity: none
Stop Bits: 1
Handshaking: none
custom BR: 9600
Rx Clear: 27
Receive: 255 Counter = 0 String
Hello world!
Hello world!
Transmit Macros: M1 500, M2 1000, M3 1000
Connected

5. Enter and run terminal BASIC program listed in **"PRG mode"** chapter below.
6. Enter something on ZX Spectrum keyboard. The entered text should appear in the Br@y++ Terminal program on the PC. The same way, you can enter text in the Br@y++ Terminal program and press **Send**. The text will appear on the ZX Spectrum screen.

Device



USB ON
(only in UART mode) both jumpers ON, if you use MINI-USB CONNECTOR. Both OFF, if you use UART CONNECTOR.

USB PWR
If device is connected to ZX Spectrum this jumper <u>MUST</u> be OFF. May be used for supply device from USB port (for example when programming firmware). Also may be used carefully to supply some ZX clones (like Harlequin), which do not consume more than 500 mA (Harlequin consumes about 250 mA).

Warning!
USB PWR is only for special purposes. Never switch this jumper ON, when working with old ('82) ZX Spectrum computer version, or ZX Spectrum is powered via external supply. You can seriously damage your computer!

INTERFACE SELECTION (1,2)
Switch between UART, I2C, SPI, anr PRG (program mode). For now, SPI does not work (is not programmed in firmware yet).

```
        00 – UART mode – UART (RS-232) or USB communication via MiniUSB or UART connector
        01 – I2C mode – I2C communication, via I2C connector
        10 – SPI mode – SPI communication, via SPI connector
        11 – PRG mode – program mode
```

CUSTOM SETTINGS
In UART mode:
3 - Set MODE:

```
        0 – TXT
        1 - HEX
```

4,5 – select speed:

```
        00 – 1200, 8,1,NO
        01 – 9600, 8,1,NO
        10 – 19200,8,1,NO
        11 – 57600,8,1,NO
```

PRG
Program mode. Sets device to terminal mode. You may set more parameters using ZX Spectrum via simple text terminal (several lines in BASIC language). You may set more baud rates, parity, stopbits, data bits, get version, etc... See chapter "PRG mode" below.

No flow control is supported. If ZX Spectrum is not receiving bytes fast enough, input buffer will overflow, and any byte incoming over the UART, that will not fit in the input buffer will be lost. The same with output buffer: if ZX Spectrum sends bytes too fast, buffer will overflow and some bytes will be lost.

TXT and HEX mode

Defines behavior for bytes incoming to ZX Spectrum, received using IN function. When receiving data with IN, you will always receive a byte. But byte 0xFF (255 dec) has special function. If you receive 0xFF it means, that there is no more bytes in input buffer, and no byte has been received.

TXT
In TXT mode, IN gives exactly byte received by UART. But this way, you cannot transfer binary data containing 0xFF bytes. It's good mode to receive text data, like terminal.

HEX
In HEX mode, each byte received in IN is converted to 3 bytes in $XX format, where XX is hexadecimal representation of received byte. In this mode, you can receive all bytes, but each byte needs to invoke IN function 3 times, and input buffer fills up three times faster.

Examples in ZX Spectrum BASIC

Send data

```
  5 LET port=163
 10 FOR x=0 TO 255
 20  PRINT AT 0,0;x;"    "
 30  OUT port,x
 40 NEXT x
 50 GOTO 10
```

Receive data

```
  5 LET port=163
 10 LET cnt=0
100 LET ch=IN port
110  IF ch<>255 THEN GO SUB 200
120 GO TO 100
200  PRINT CHR$(ch);
210  LET cnt=cnt+1
220  IF cnt=640 THEN CLS : LET cnt=0
299 RETURN
```

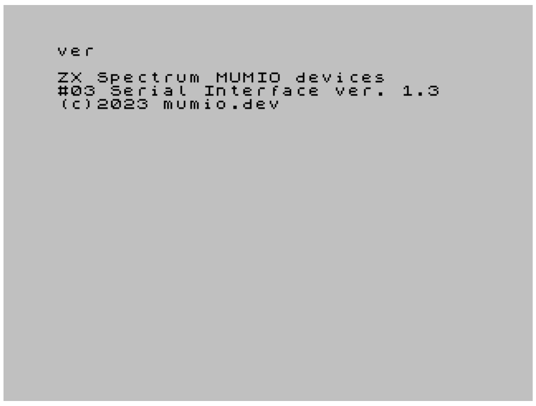
PRG mode

In PRG mode, you can communicate with Serial Interface device, using ZX Spectrum as character terminal. In this mode, you may ask device for version, current config params, set config params, and something more.

Just write and run simple terminal BASIC program, and enter some commands: **ver**, **get**, **set**, **wopr**, **help**...

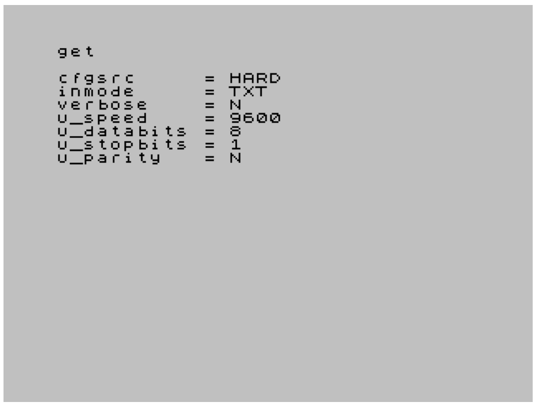
```
  2 REM #######################
  3 REM #  Serial terminal    #
  4 REM #######################
  5 LET port=163
  7 REM #######################
 10 LET a$=INKEY$: GO SUB 70
 20 IF a$<>"" THEN GO TO 10
 30 LET a$=INKEY$: GO SUB 70
 40 IF a$="" THEN GO TO 30
 50 PRINT a$;: OUT port,CODE a$
 60 GO TO 10
 65 REM #######################
 70 LET ch=IN port
 80 IF ch<>255 THEN PRINT CHR$(ch);
 85 POKE 23692,255
 90 RETURN
 95 REM #######################
100 SAVE hd0"/bas/term.bas"
```

Using the terminal, you can communicate with the device, in particular change configuration parameters that are not available from the switches. The most important parameter "cfgsrc" determines whether the software parameters set here are to be used and the switch settings are to be ignored (SOFT) or vice versa (HARD).



```
ver

ZX Spectrum MUMIO devices
#03 Serial Interface ver. 1.3
(c)2023 mumio.dev
```



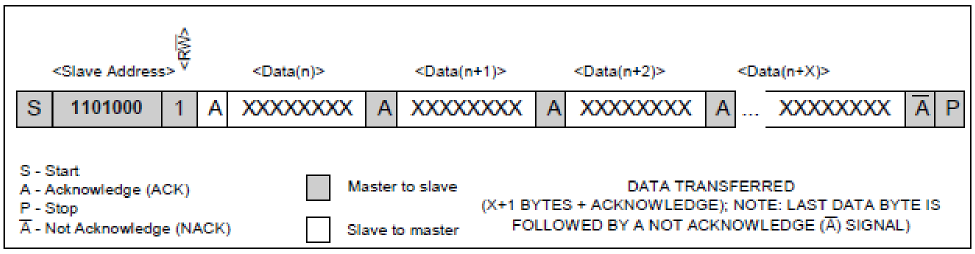
```
get

cfgsrc      = HARD
inmode      = TXT
verbose     = N
u_speed     = 9600
u_databits  = 8
u_stopbits  = 1
u_parity    = N
```

I^{2}C mode

If you read the documentation for I2C devices, you will notice that in the vast majority of cases, communication takes place in small packets, among which we distinguish three basic ones: Read, Write and Write/Read.

Data Read



Is serviced via **Read (R)** frame described below. First, you need to send 5 bytes with the following structure using the OUT command:

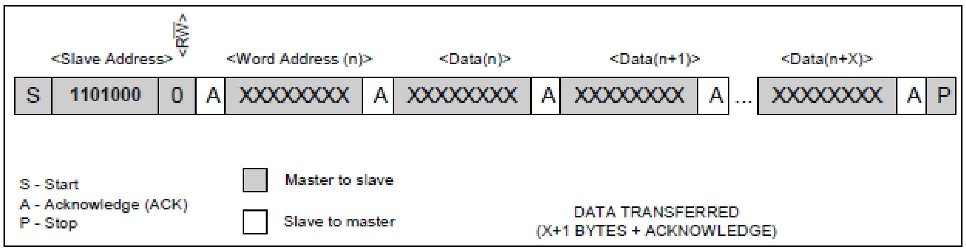
```
        243 AD RH RL 241
```

...and receive, using IN command, RH*256+RL bytes response.

```
   OUT:
   00 243 (0xF3) - CTL_I2C_R
   01 <address> (7 bits)
   02 <read_len_H>
   03 <read_len_L>
   04 241 (0xF1) - CTL_EOF

   IN:
   <read_byte..1>
   <read_byte..2>
   <read_byte..n>
```

Data Write



Is serviced via **Write (W)** frame described below. First, you need to send at least 6 bytes with the following structure using the OUT command:

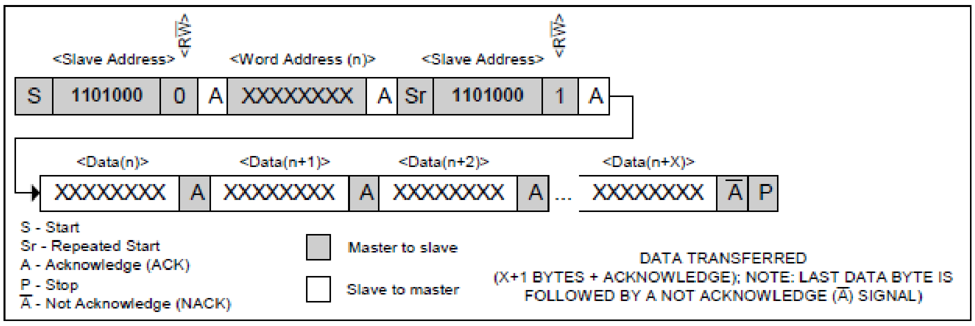
```
       244 AD WH WL d1..dn 241
```

...and receive, using IN command, one byte return value response.

```
   OUT:
   00 244 (0xF4) - CTL_I2C_W
   01 <address> (7 bits)
   02 <write_len_H>
   03 <write_len_L>
   04 <write_byte..1>
   05 <write_byte..2>
   06 <write_byte..n>
   xx 241 (0xF1) - CTL_EOF

   IN:
   Return value (0-ok, 1-fail)
```

Data Write and Read



Is serviced via **Write And Read (R)** frame described below. First, you need to send at least 8 bytes with the following structure using the OUT command:

```
        245 AD WH WL RH RL d1..dn 241
```

...and receive, using IN command, RH*256+RL bytes response.

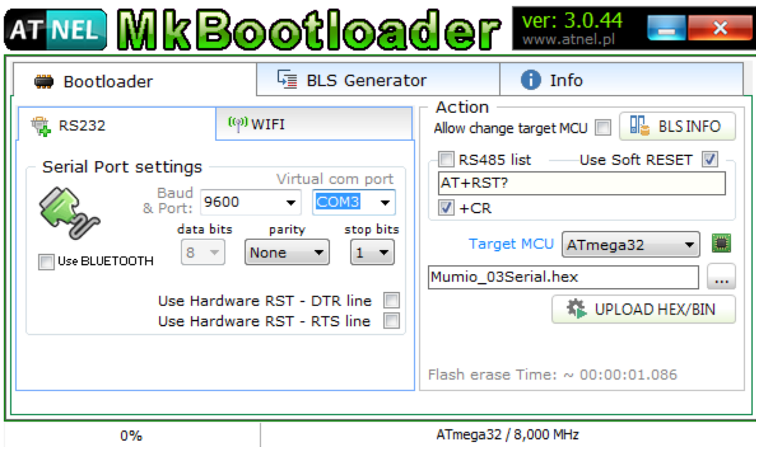
```
   OUT:
   00 245 (0xF5) - CTL_I2C_WR
   01 <address> (7 bits)
   02 <write_len_H>
   03 <write_len_L>
   04 <read_len_H>
   05 <read_len_L>
   06 <write_byte..1>
   07 <write_byte..2>
   08 <write_byte..n>
   xx 241 (0xF1) - CTL_EOF

   IN:
   <read_byte..1>
   <read_byte..2>
   <read_byte..n>
```

```
Descriptions:
AD - I2C address (7 bits only)
WH - write data length hi byte
WL - write data length lo byte
RH - read data length hi byte
RL - read data length lo byte
d1..dn - data bytes
```

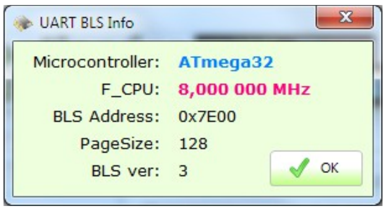
Firmware update

1. Disconnect device from ZX Spectrum edge connector, and dissasemble Serial Port (#03) from I/O Port (#00). Device should be standalone, connected to nothing.
2. Install all three jumpers: USB ON and USB PWR. The USB will be connected to CPU and board will be powered by USB.
3. Connect PC via USB cable.
4. Run MkBootloader program.



Serial Port settings
Baud & Port: 9600
Virtual com port: COM3
data bits: 8
parity: None
stop bits: 1
Use BLUETOOTH
Use Hardware RST - DTR line
Use Hardware RST - RTS line
Allow change target MCU
RS485 list
Use Soft RESET ✓
AT+RST?
✓ +CR
Target MCU: ATmega32
Mumio_03Serial.hex
Flash erase Time: ~ 00:00:01.086
0%
ATmega32 / 8,000 MHz

5. Select Virtual com port
6. Press BLS INFO. MkBootloader will establish communication with the CPU and display its version. If connection cannot be established - reset CPU by removing and inserting USB PWR jumper. You have only 5 seconds since reset to establish bootloader communication!



Microcontroller: ATmega32
F_CPU: 8,000 000 MHz
BLS Address: 0x7E00
PageSize: 128
BLS ver: 3

7. Select target MCU to Atmega32 and select firmware program (Mumio_03Serial.hex).
8. Press UPLOAD HEX/BIN - firmware program will be uploaded to device.
9. Remove USB PWR jumper!
10. Connect Serial Port (#03) to I/O Port (#00) and connect them to ZX Spectrum edge connector.