# MASS/DIMM instrument

## Reprogramming of the electronics for using the SLIP protocol

V. Kornilov

May 2, 2016

## 1 Introduction

The combined MASS-DIMM instrument has been developed in 2003–2004 in anticipation of the large-scale site-testing campaigns for very large optical telescopes.

The MASS part of the instrument is a fast 4-channel photometer measuring light fluxes with a temporal resolution of 1 ms or better. The data are sent to the control computer through a serial link using physical RS485 interface. The fast exchange speed is acheved by sending the data in binary format with a speed of 460 to 1840 kB/s. A custom logical protocol for the data exchange has been developed. It is described in `http://curl/mass/download/doc/new_details.pdf` and `http://curl/mass/download/doc/new_details_v2.pdf`. These documents also contain the detailed design of the MASS electronics. The protocol uses the 9-th bit of a transmitted symbol to mark the start of the data packet. This is a standard technique that can be easily implemented in the microcontroller. However, a special driver is needed in the computer, on the receiving end. The problem was partly solved by developing a custom electronic module for LPR/RS485 interface.

By 2010 it became evident that personal computers replace parallel port with USB. Development of a different interface to the MASS instrument was therefore needed. Experience indicated that supporting a custom interface with a special driver is too labor-intensive, considering about 30 MASS instruments scattered all over the world. Availability of commercial computers with fast serial ports and of standard USB/RS485 convertors working in harsh environment conditions with a speed up to 1–3 Mb/S made things easier. However, the MASS data exchange protocol prevented the use of these convertors, requiring switching of the 9th bit and having a narrow time window for the data packets (250–500 $\mu$s).

To solve the problem, we revised the data exchange protocol. Instead of the 9th bit, a special symbol delimiting the data packets is now used. The structure of the data packets remains unchanged, following the encapsulation principle inherent to the Serial Line Internet Protocol (SLIP). However, the change of the protocol requires reprogramming of the MASS electronics with the help of an external programmator, because the existing reprogramming utility avrup cannot change the boot segment of the microcode.

## 2 Brief description of the SLIP rotocol

The basic idea of the SLIP protocol is transmission of binary data as logically organized data structures, or datagrams. Its description for the case of internet connection can be found in Wikipedia `https://en.wikipedia.org/wiki/Serial_Line_Internet_Protocol`. The start (and/or end) of a datagram is marked by a special symbol `END`. The uniqueness of this symbol is achieved by byte staffing inside the data packet by the escape-sequence, where the byte `END (0x0)` is replaced by the sequence (`0xDB, 0xDC`) and the byte `ESC (0xDB)` is replaced by (`0xDB, 0xDD`).

The byte staffing is done after the full data packet of the MASS protocol is formed. Then the packet is transmitted, and the reverse transformation, byte unstaffing, is done on the receiving end. These transformations are done in the MASS electronics "on the fly", and then the control sum of the data packet `CRC` is checked. In the control computer, these transformations are done in the software, in the code module exchange.cpp. Serial data transmission uses standard serial drivers `/dev/ttyS0...  /dev/ttyUSB0...`.

The new protocol still has information signals (special 1-byte packets), but now such signals are generated only by the electronics modules and are also terminated by the symbol `END`.

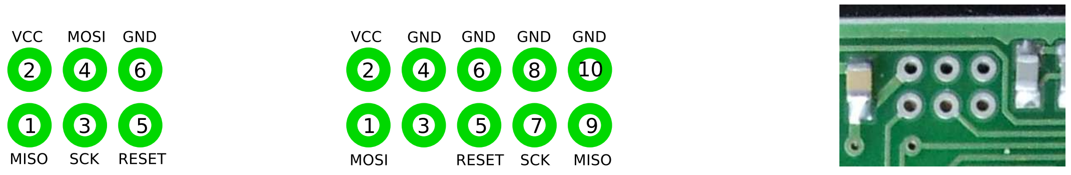

Figure 1: Pin layout of ISP6 and ISP10 connectors, looking top down to the PCB. Image on the right shows part of the MASS power module with ISP sockets, oriented in the same way as the ISP6 connector on the left.

Acknowledgement of the data reception by the computer by the signal `ACK (NAK)` has been dropped. The reason is the difficulty to ensure the timely response from the software. When the control sum does not match, repeated data transmission is requested by the computer.

The data packet transmitted by a module now also contains its address. The average number of transmitted symbols has now increased somewhat, but each symbol became shorter (8 bits instead of 9), so for long data packets there is actually a gain.

The protocol was tested using the USB/RS485 convertor with FTDI chip with exchange speed of 960 Baud. In the Linux OS it was registered as device `/dev/ttyUSB0`. In the command regime of requests and answers the exchange cycle takes less than 1 $\mu$s even when large 32-byte data are transmitted. This is slightly faster than with the original protocol, because the FTDI chip has a large internal FIFO (128 bytes), making large extra pauses between transmissions no longer necessary. Testing in the active regime, when the data transmission to the computer is initiated by the counters, have shown that the 32-byte data packets (16 readings of the counters) allow work with a micro-exposure as short as 0.2 ms.

## 3 Tools for reprogramming

To reprogram fully the microcontrollers ATmega8 with flash memory used in the MASS electronic modules, any device for the In-System Programming (ISP) can be used, either from the same manufacturer Atmel or from other companies. To this end, the MASS electronic boards have sockets corresponding to the ISP6 standard (see Fig. 1). The only difference from the standard is the pitch of 2.0 mm instead of the normal 2.54 mm. Therefore, a special interface adapter is needed for coupling either to the ISP6 or to the ISP10 standard connectors.

We used mostly the simplest programmator STK200/300 shown in Fig. 2 that connects to the parallel port of the computer. The programmator is powered by the device being programmed, while the required sequence of ISP signals is generated by the control program. A large variety of software tools for ISP programming is available. We found that the free cross-platform console program `avrdude` is most convenient. It is developed for programming Atmel microcontrollers of AVR series, see `http://www.nongnu.org/avrdude/`.

This document is accompanied by the description of `avrdude`. It can be installed on any Linux distribution using the system packet manager. By default the device `/dev/parport0` will be used.

Programming in correct order is done by the shell scripts `bic1.sh`, `bic2.sh`, and `massdimm.sh`. They program sequentially the fuse bits of the corresponding microcontroller, the load the boot block into the flash memory, and finally load the functional microcode.

To check correct installation of `avrdude`, connect the programmator to the parallel port and type `avrdude -p m8 -c stk200`. The following message should appear:

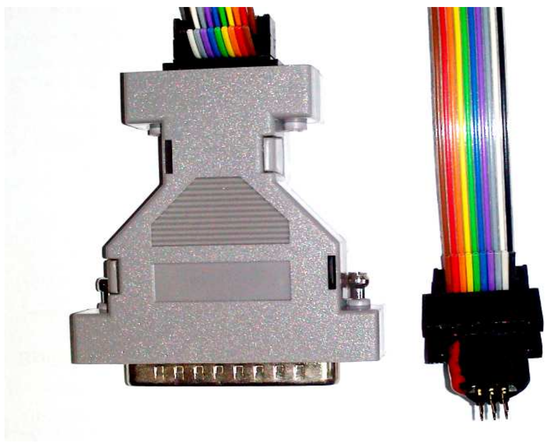

Figure 2: ISP programmator STK200 (its custom-made analog). On the right, the adapter from the ISP10 connector to ISP6 sockets with a 2.0-mm pitch is shown.

```
avrdude: AVR device not responding
avrdude: initialization failed, rc=-1
         Double check connections and try again, or use -F to override
         this check.

avrdude done.  Thank you.
```

This means that `avrdude` works but target microcontroller does not respond. During programming, the messages may appear on the screen. Most frequent reasons for a failure are 1) no power in the device being programmed, 2) bad contact between the sockets of the device and the programmator interface connector, and 3) connection of the adaptor turned by 180°. In such cases, three these messages appear sequentially.

The inverted connection does not damage the electronics because the programmator has a hardware protection against this.

## 4 Reprogramming

Reprogramming of the MASS electronics does not require its disassembly. Just remove the covers of the PMT section and of the power module. Looking at the counter modules with the mechanical attachments facing up, as shown in Fig. 3 , we see the sockets of `bic2` on the left and the sockets of `bic1` on the right. The ISP contacts are oriented in such way that the adaptor must be connected with the red band on its right side.

Connect the adaptor to the sockets of `bic1`, turn the power of the MASS electronics on, slightly press the adaptor to ensure a good contact, and run the script `bic1.sh` from the working directory `./bic1.sh`. During the reprogramming process, information on the progress and on the verification of the loaded microcode is displayed:

```
victor@photon:~/AVRprogramm/slip/mass> ./bic1.sh

avrdude: AVR device initialized and ready to accept instructions
```

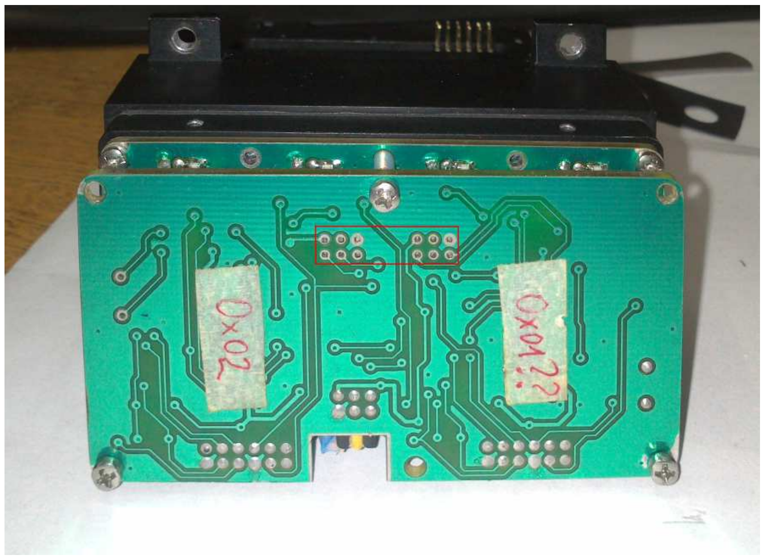

Figure 3: Counters circuit board with the 2-mm pitch ISP sockets (in the red box). The sockets of `bic2` are on the left, the sockets of `bic1` on the right.

```
Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e9307
avrdude: reading input file "0x3f"
avrdude: writing lfuse (1 bytes):

Writing | ################################################## | 100% 0.00s

avrdude: 1 bytes of lfuse written
avrdude: verifying lfuse memory against 0x3f:
avrdude: load data lfuse data from input file 0x3f:
....................
....................
....................
avrdude: input file Bicounter1.hex contains 1618 bytes
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 0.52s

avrdude: verifying ...
avrdude: 1618 bytes of flash verified

avrdude: safemode: Fuses OK

avrdude done.  Thank you.
```

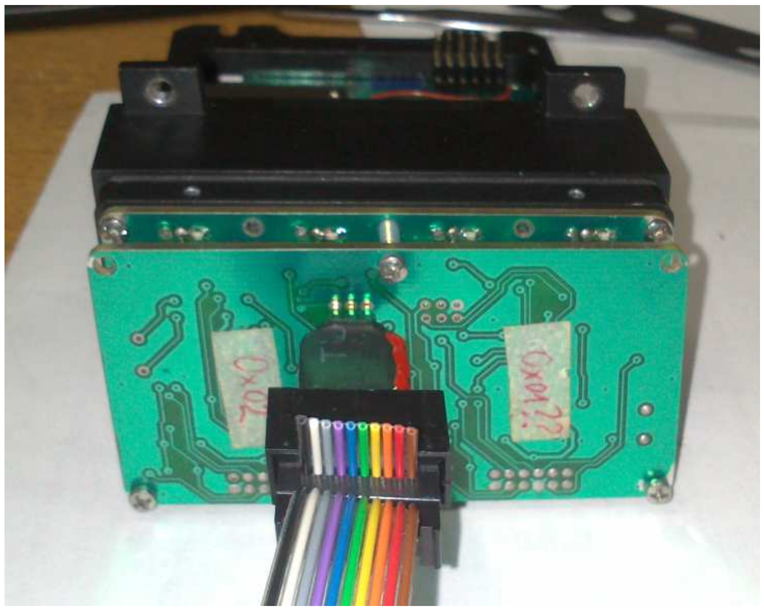

Figure 4: Connection of the programming adaptor to then sockets of `bic2`.

When the reprogramming is finished, turn the MASS power off, disconnect the adapter and plug it into `bic2`, as shown in Fig. 4. Power the MASS again and run the script `bic2.sh`. Then turn the MASS power off and disconnect the adapter.

When the power module is reprogrammed, be careful not to shorten the pins of the adapter by the metal frame of the electronics box, located just beneath the sockets. To prevent this, loosen the screws that fix the PCB to the frame and insert a thin leaf of insulation below the ISP sockets. Orient the power module as shown in Fig. 5 and connect the adapter, red band on the right side. Turn the MASS power on and run the script `massdimm.sh`. When done, turn the MASS power off and disconnect the adapter.

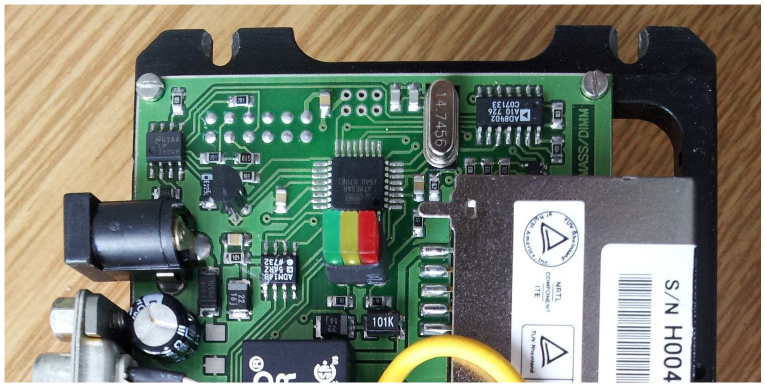

Figure 5: In this orientation of the power module the pins of the ISP socket correspond to Fig. 1.

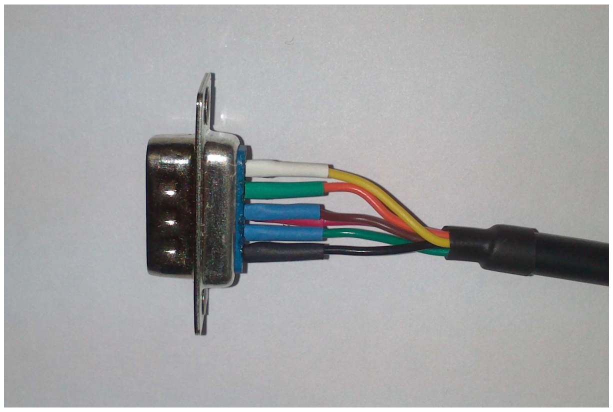

Figure 6: Correct connection of the original FTDI USB/RS485 cable wires to DB9M connector. The black wire is connected to pin 5 of the DB9M, the yellow wire — to pin 1, the orange wire — to pin 2. Other wires non needed but are connected to prevent accidental short-circuiting to a free pins of the DB9M: the red (+5) — to pin 8, the brown (terminator) to pin 3, and the geen wire is connected to pin 4.

## 5 Testing communication between computer and MASS

After reprogramming, the MASS electronics must be tested. First, replace the cover of the PMT module and verify that the PMTs are closed (during the test, high voltage may be accidentally turned on). Connect MASS to the USB port using the FTDI USB/RS485 convertor. Correct connection of the original FTDI cable is shown in Fig. 6. Other convertor cables circulated on market can have four wires only in different colors.

The command `dmesg` can be used to find the number of the USB port assigned by the OS to the convertor. After reprogramming, MASS uses the communication speed of 920 Kb/s. Corresponding modifications are needed in the configuration file `mass.cfg`, for example

```
Device       /dev/ttyUSB0  ;
BaudeRate             920  ;
```

The identifiers of all three modules have changed after reprogramming. Therefore, first run the program as `mass -d -a`, to see the initialization commands on the screen. Responses to the command `a2` will tell us the new device identifiers, to be written in the config file.

Using `telnet` connection to the port 16201, run the exchange test by the command `run etest`. Further tests must be done after installing the electronics in the MASS-DIMM instrument.