AVR Simple serial port programming

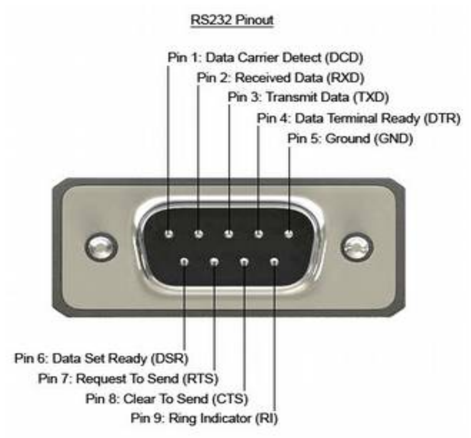
release 1.1

I wanted to test a simple AVR programming interface using a serial port with bit banging (no RS232 official signals)

Programmation software :
I use the widespread **avrdude** software in Linux environment (flavor Mint 17)
Avrdude allows two serial port bitbang programming options

-c dasa :
RESET = RTS (pin 7, connected to pin 6 DSR)
SCK = DTR (pin 4)
MOSI = TxD (pin 3)
MISO = CTS (pin 8)

-c dasa3 :
RESET =!DTR (pin 4) (reset is inverted)
SCK = RTS (pin 7, connected to pin 6 DSR)
MOSI = TxD (pin 3)
MISO = CTS (pin 8)


RS232 Pinout
Pin 1: Data Carrier Detect (DCD)
Pin 2: Received Data (RXD)
Pin 3: Transmit Data (TXD)
Pin 4: Data Terminal Ready (DTR)
Pin 5: Ground (GND)
Pin 6: Data Set Ready (DSR)
Pin 7: Request To Send (RTS)
Pin 8: Clear To Send (CTS)
Pin 9: Ring Indicator (RI)

DTE=male
pin side view

DCD	= in
RxD	= in
DSR	= in
CTS	= in
TxD	= out
DTR	= out
RTS	= out

picture from : http://www.usconverters.com/index.php?main_page=page&id=61&chapter=0

the port should be specified :
- in the avrdude.cont as default serial (/dev/ttyUSB0)
- or in the command line (-P /dev/ttyUSB0)

complete command line
for terminal mode :
sudo avrdude -p t13 -c dasa3 -i 5000 -P /dev/ttyUSB0 -v -t
for programmation :
sudo avrdude -p t13 -c dasa3 -i 5000 -P /dev/ttyUSB0 -v -U flash:w:main.hex

The programming and verification take one minute for 114 bytes in a tiny13A

Hardware

The hardware is designed to never exceed the Atmel specs. Atmel states that the voltage at any pin sould not exceed the power voltage + 0,5V or the ground voltage -0,5V. Obviously, a simple diode clamping could not meet these specs, as it has a Vd at 0,6V or more. If a Zener is used, it should be from accuracy high enough AND has to match the Vcc, anyway it allows the voltage to go under GND-0,5V limit.
That is why I choose to clamp the signals to a resistor of 100 ohms (to drain enough current, and HF decoupled by a 10nF capacitor). This resistor is connected to Vcc and GND by 2 Silicon diodes. The clamping diodes are connected to the 100 ohms resistor, so the pins voltage will never exceed Vcc and GND, even if Vcc is not precisely 5V, as it is allowed to go up to 6V. As long as the signals levels are inbetween Vcc and GND, the impedance of the line is very high,

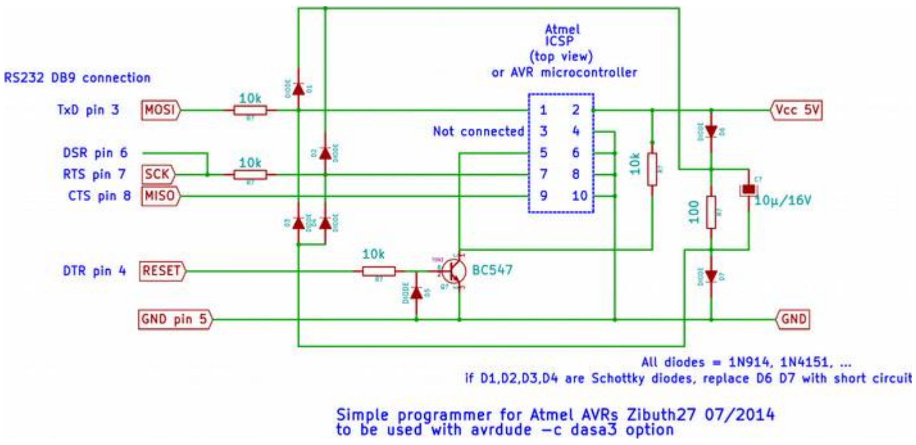
limited by the AVR only.

The hardware is different whether the programmer uses - dasa or – dasa3 avrdude command.
Not only the RS232 pins are different, but the RESET signal is also inversed, which requires a transistor for the dasa3 option.

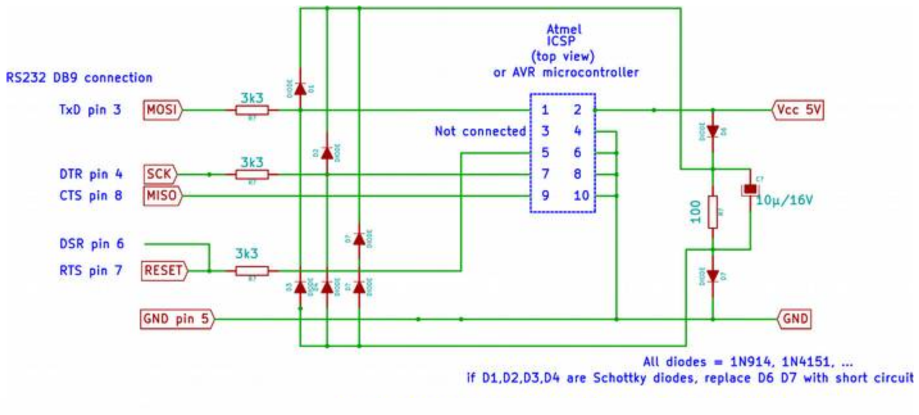
For serial port I used a USB-RS232 cable with a DB9 connector, because a small USB-RS232 card generally has only 0 to 5V TxD and RxD signals. This programming interface is able to work with any standard RS232 signals up to +-20V. This interface is recognized as /dev/ttyUSB0 in Linux systems and shoud be declared in avrdude.conf or in the command line.

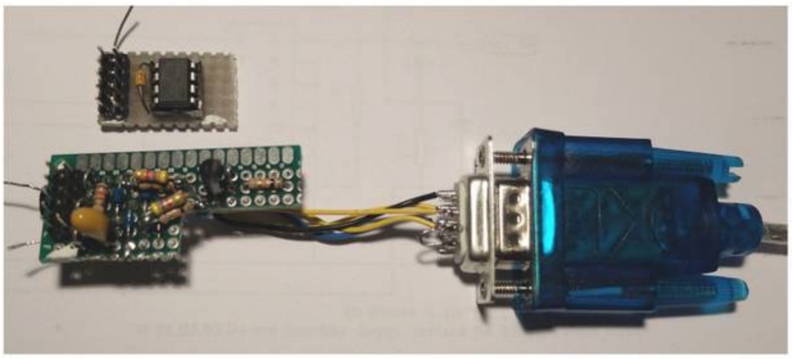
The resistors in series with RS232 connector in both dasa & dasa3 can go down to 1k, provided the AVR has no connections with any load (on socket used only for programmation)

For -*dasa3* option

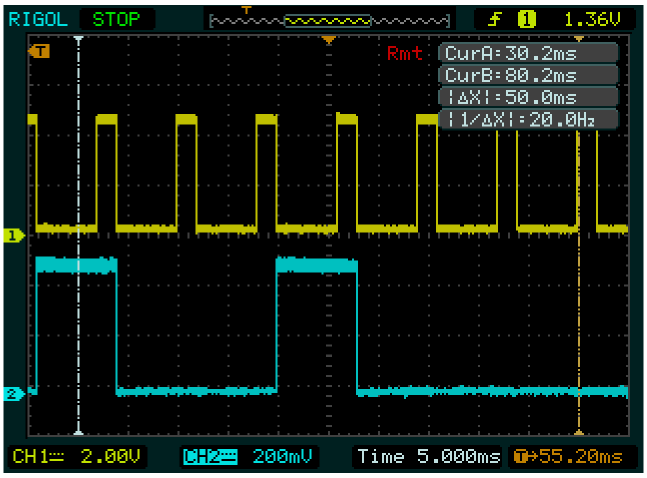


for -*dasa* option

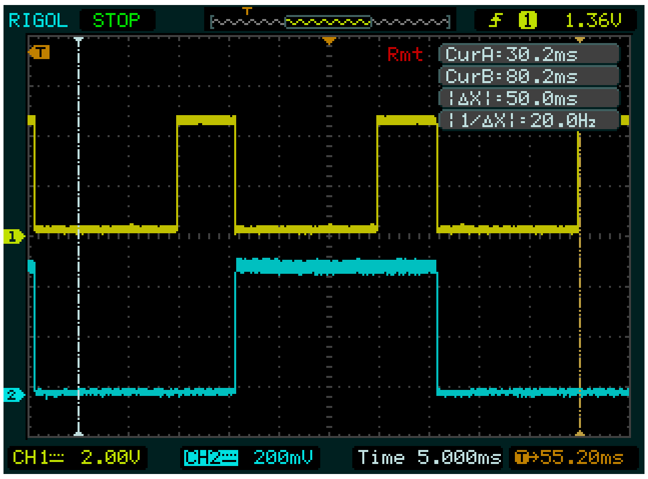




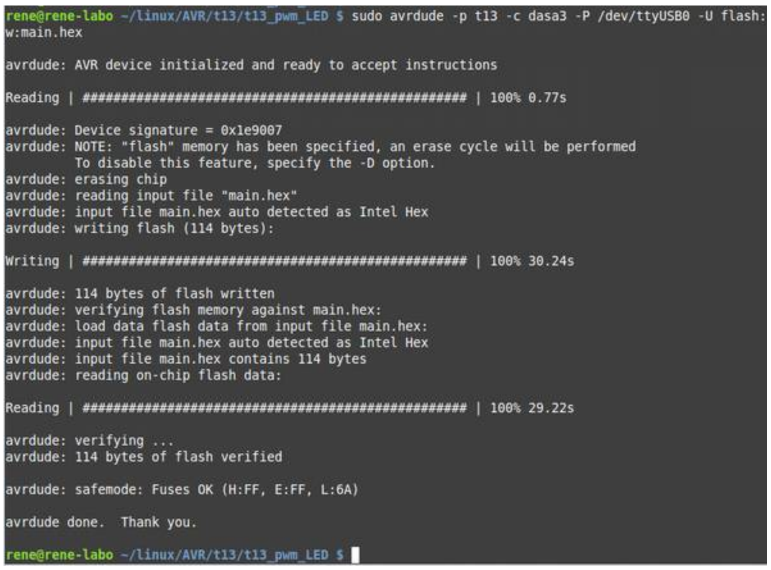
the dasa3 programmer board
with tiny13 socket
and RS232 – USB cable
ICSP 10pin links



dasa3 SCK MOSI



dasa3 with -i 5000 option



```
rene@rene-labo ~/linux/AVR/t13/t13_pwm_LED $ sudo avrdude -p t13 -c dasa3 -P /dev/ttyUSB0 -U flash:
w:main.hex

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.77s

avrdude: Device signature = 0x1e9007
avrdude: NOTE: "flash" memory has been specified, an erase cycle will be performed
         To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "main.hex"
avrdude: input file main.hex auto detected as Intel Hex
avrdude: writing flash (114 bytes):

Writing | ################################################## | 100% 30.24s

avrdude: 114 bytes of flash written
avrdude: verifying flash memory against main.hex:
avrdude: load data flash data from input file main.hex:
avrdude: input file main.hex auto detected as Intel Hex
avrdude: input file main.hex contains 114 bytes
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 29.22s

avrdude: verifying ...
avrdude: 114 bytes of flash verified

avrdude: safemode: Fuses OK (H:FF, E:FF, L:6A)

avrdude done.  Thank you.

rene@rene-labo ~/linux/AVR/t13/t13_pwm_LED $
```

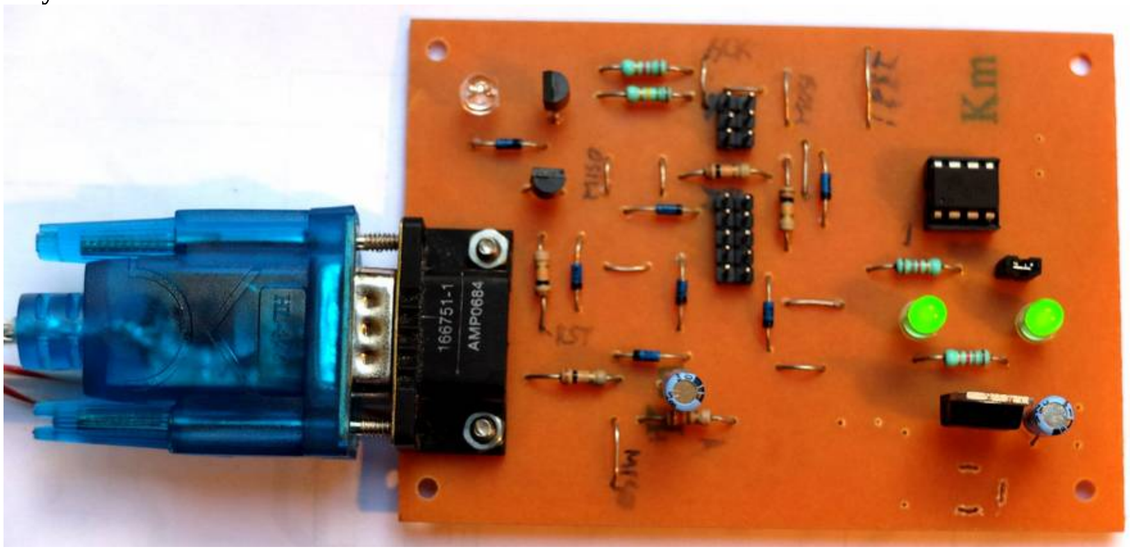
Results

The dasa3 option works well !
The -i option allows to program slow µcontrollers (for slow internal clock programmed MCUs)

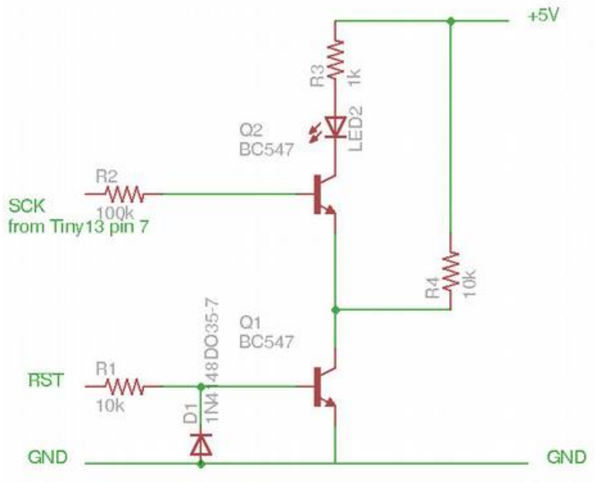
The -c dasa option does not work at all, I contacted Jörg Wunsch (last author of avrdude) to know if it might be a bug or simply if -c dasa might no longer be supported. No answer, one month later.

Of course, this programmer is pin-to-pin compatible (same cable) with standard programmers like USBASP

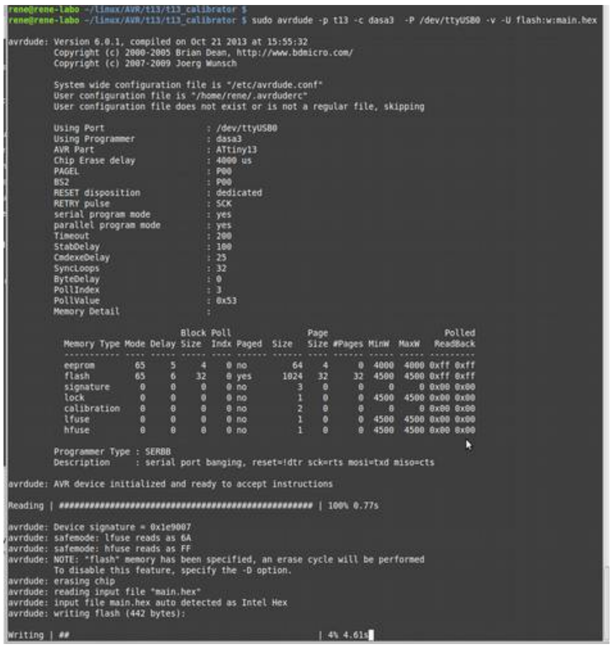
PCB made by Yoruk



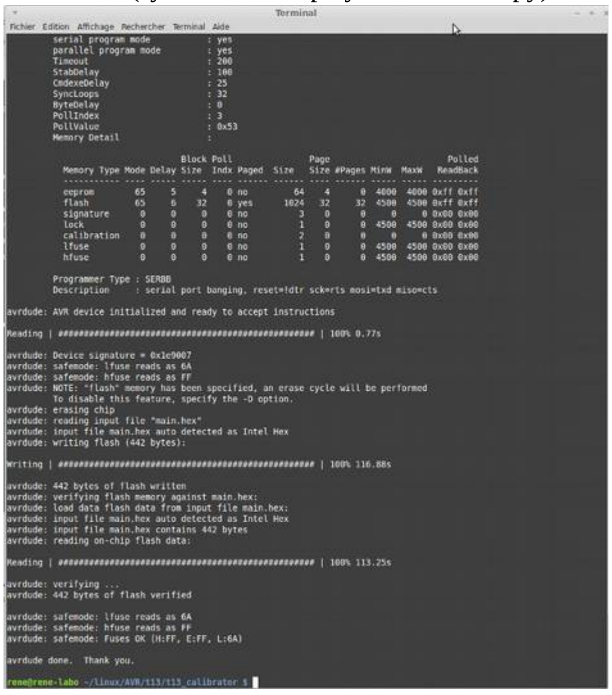
Yoruk made this PCB and added a « programming in process » LED, following my indications



Everything works ! (including prog_LED), programmed with avrdude (syntax used displayed on hard-copy)



```
rene@rene-labo ~/linux/AVR/t13/t13_calibrator $
rene@rene-labo ~/linux/AVR/t13/t13_calibrator $ sudo avrdude -p t13 -c dasa3  -P /dev/ttyUSB0 -v -U flash:w:main.hex

avrdude: Version 6.0.1, compiled on Oct 21 2013 at 15:55:32
         Copyright (c) 2000-2005 Brian Dean, http://www.bdmicro.com/
         Copyright (c) 2007-2009 Joerg Wunsch

         System wide configuration file is "/etc/avrdude.conf"
         User configuration file is "/home/rene/.avrduderc"
         User configuration file does not exist or is not a regular file, skipping

         Using Port                    : /dev/ttyUSB0
         Using Programmer              : dasa3
         AVR Part                      : ATtiny13
         Chip Erase delay              : 4000 us
         PAGEL                         : P00
         BS2                           : P00
         RESET disposition             : dedicated
         RETRY pulse                   : SCK
         serial program mode           : yes
         parallel program mode         : yes
         Timeout                       : 200
         StabDelay                     : 100
         CmdexeDelay                   : 25
         SyncLoops                     : 32
         ByteDelay                     : 0
         PollIndex                     : 3
         PollValue                     : 0x53
         Memory Detail                 :

                                  Block Poll               Page                       Polled
           Memory Type Mode Delay Size  Indx Paged  Size   Size #Pages MinW  MaxW   ReadBack
           ----------- ---- ----- ----- ---- ------ ------ ---- ------ ----- ----- ---------
           eeprom        65     5     4    0 no         64    4      0  4000  4000 0xff 0xff
           flash         65     6    32    0 yes      1024   32     32  4500  4500 0xff 0xff
           signature      0     0     0    0 no          3    0      0     0     0 0x00 0x00
           lock           0     0     0    0 no          1    0      0  4500  4500 0x00 0x00
           calibration    0     0     0    0 no          2    0      0     0     0 0x00 0x00
           lfuse          0     0     0    0 no          1    0      0  4500  4500 0x00 0x00
           hfuse          0     0     0    0 no          1    0      0  4500  4500 0x00 0x00

         Programmer Type : SERBB
         Description     : serial port banging, reset=!dtr sck=rts mosi=txd miso=cts

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.77s

avrdude: Device signature = 0x1e9007
avrdude: safemode: lfuse reads as 6A
avrdude: safemode: hfuse reads as FF
avrdude: NOTE: "flash" memory has been specified, an erase cycle will be performed
         To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "main.hex"
avrdude: input file main.hex auto detected as Intel Hex
avrdude: writing flash (442 bytes):

Writing | ##                                                 | 4% 4.61s
```



Terminal

Fichier Edition Affichage Rechercher Terminal Aide

```
         serial program mode           : yes
         parallel program mode         : yes
         Timeout                       : 200
         StabDelay                     : 100
         CmdexeDelay                   : 25
         SyncLoops                     : 32
         ByteDelay                     : 0
         PollIndex                     : 3
         PollValue                     : 0x53
         Memory Detail                 :

                                  Block Poll               Page                       Polled
           Memory Type Mode Delay Size  Indx Paged  Size   Size #Pages MinW  MaxW   ReadBack
           ----------- ---- ----- ----- ---- ------ ------ ---- ------ ----- ----- ---------
           eeprom        65     5     4    0 no         64    4      0  4000  4000 0xff 0xff
           flash         65     6    32    0 yes      1024   32     32  4500  4500 0xff 0xff
           signature      0     0     0    0 no          3    0      0     0     0 0x00 0x00
           lock           0     0     0    0 no          1    0      0  4500  4500 0x00 0x00
           calibration    0     0     0    0 no          2    0      0     0     0 0x00 0x00
           lfuse          0     0     0    0 no          1    0      0  4500  4500 0x00 0x00
           hfuse          0     0     0    0 no          1    0      0  4500  4500 0x00 0x00

         Programmer Type : SERBB
         Description     : serial port banging, reset=!dtr sck=rts mosi=txd miso=cts

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.77s

avrdude: Device signature = 0x1e9007
avrdude: safemode: lfuse reads as 6A
avrdude: safemode: hfuse reads as FF
avrdude: NOTE: "flash" memory has been specified, an erase cycle will be performed
         To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "main.hex"
avrdude: input file main.hex auto detected as Intel Hex
avrdude: writing flash (442 bytes):

Writing | ################################################## | 100% 116.88s

avrdude: 442 bytes of flash written
avrdude: verifying flash memory against main.hex:
avrdude: load data flash data from input file main.hex:
avrdude: input file main.hex auto detected as Intel Hex
avrdude: input file main.hex contains 442 bytes
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 113.25s

avrdude: verifying ...
avrdude: 442 bytes of flash verified

avrdude: safemode: lfuse reads as 6A
avrdude: safemode: hfuse reads as FF
avrdude: safemode: Fuses OK (H:FF, E:FF, L:6A)

avrdude done.  Thank you.

rene@rene-labo ~/linux/AVR/t13/t13_calibrator $
```