ATMEGA Programmer

Autor:
Data de publicació: 12-07-2016

We program MCUs

Content

1 Introduction
2 Description of single-chip computer Atmega8

2.1 Input Output ports

3 What we need
4 program ATmega through the parallel port

4.1 Production of a simple programmer STK200
4.2 Compiling a program in assembler flashing LEDs
4.3 Load program into ATMega
4.4 Switching fuses
4.5 Facilitating work with Linux avrdude

5 Program counting pulses
6 Use input / output ports
7 Using the A / D converters

7.1 Measurement Specification

8 Writing and reading internal EEPROM memory
9 Overclock at a higher frequency with the crystal
10 Using interruptions
11 Communication with PC through USART

11.1 Hardware
11.2 Firmware ATMega to communicate via USART
11.3 Software on your computer

11.3.1 Linux Command Line
11.3.2 Program in Python

11.4 advanced communication solutions through a serial port

12 Recording to SD memory cards
13 Communication with PC via USB
14 Construction USB Programmer
15 Appendices

15.1 Short glossary
15.2 High Voltage Parallel Programming

15.2.1 Motivation
15.2.2 Materials Required
15.2.3 Program Uploading doctor
15.2.4 How to fix fuse

15.3 To write the parallel port in Linux

16 Useful links

Introduction

The book assumes a slightly advanced knowledge of programming (in languages ??C and Python), electronics , hexadecimal and computing in general. Examples of these are written in two languages ??and is expected to Linux environment on a PC (instructions tested on Ubuntu 8.04). If you have Windows or any other system, probably the situation does not differ much with minor modifications, can be examples in this book use as well.

The entire book is for the purpose of simplicity will only deal with one type of single-chip computer, and it Atmega8 and a newer version of ATmega8. In the store you buy it in a matter of 50 to 70 CZK under the name ATMEGA8. The book is conceived in a "jump into the water and learn to swim." For any confusion with readers we apologize and hope that the missing parts conjecture or find elsewhere. Examples are commented on them, and if something is not completely clear, we will try to bring enough references. Instead of the classical schemes are sometimes used directly sketches chips from top to make it clear location pins.

Do not be afraid to talk to ascribe this page your questions, comments and requests to the next chapter. Text is in an early stage of development. You can also contribute with their experience.

Description of single-chip computer Atmega8

Atmega8 in a standard 28-leg DIP case

Atmega8 is called. Single-chip computer or microcontroller. The single chip contains all of the components that are needed to run a computer, even if they are compared with a desktop PC rather weak:

AVR microcontroller

the eight-bit (normal PC is 32-bit)
It has called. RISC architecture, which is different from the architecture commonly used processors Intel or AMD
It operates at a frequency of 1 MHz, but it can overclock up to 16 MHz

8KB In-System-Programmed FLASH memory

This memory is used to record your own program, is retained even after shutdown

1kB RAM

freely accessible memory to the operating data (tripping deletes)

512B EEPROM

durable memory for long-term storage of operational data

other embedded circuits

2x + 1x16bit 8bit timer / counter
3 PWM - circuits St?ídová modulation (control of DC motors, etc.).
4x 10-bit A / D, 2x 8-bit A / D - 6 digital voltmeters (1024 or 256 values)
USART - communication via the serial port
internal oscillator 1-8 MHz - serves as a source of clock frequency, if not overclocked chip external crystal or RC circuit
watchdog timer - automatic reset in case of freezing program

Supply of the entire single-chip computer is + 5 V. Current consumption is small at a frequency of 1 MHz is the order of 3 mA. The consumption is proportional to the frequency and it can significantly reduce podtaktováním e.g. at 1 kHz. Note that 8 KB of program memory is completely separate from one kilobyte of RAM and running ATMega into them on paper. This is called the Harvard architecture. In districts ATMega is more I / Os than the chip legs. It is believed that never use any internal pins simultaneously. The individual legs during programming can connect to a given inner conductor and thereby determine their function. The following chapters take a closer look at the features and capabilities of the chip and the functional interpretation illustrated instructions.

Input output ports
Atmega8 has three ports: two eight-bit PORTB, PORTD and one six-bit PORTC. Their individual pins can operate as an input or output. That will be determined individually for each pin registers DDRA, DDRB, DDRC.

For most pins for more labels. These pins can be run for the microcontroller to switch between whether to behave like a digital port, or whether they will use a special feature that they have been assigned. (The exception is pin RESET / PC6, which can be changed to an I / O pin setting fuses RSTDSBL only during programming.) The list of special features:

RESET - reset the microcontroller, required for serial programming
RXD, TXD - receiving and sending data via the serial interface
INT0, INT1 - two sources of external interrupts
T0, T1

XCK
TOSC1, TOSC2
XTAL1, XTAL2 - the ability to connect an external crystal
AIN0, AIN1 - input analog comparator
ICP1
OC1A, OC1B, OC2
MISO, MOSI, SCK - automatically used during serial programming (data transfer inside out and clocking)
SCL
SDA
ADC0, ADC1, ADC2, ADC3 - four 10-bit analog-to-digital converters (ADC)
ADC4, ADC5 - two 8-bit ADC

What we need

Instructions assume that the reader has a computer with one hand parallel port (If none of the computers Netrc, do not worry, there are still frequent internal connector on the motherboard -labeled as LPT- outlet and comes to CZK 80), both the basic electronic equipment such as wires, soldering like. Specific needs depends on the use ATMega. If readers want to start with the first few instructions, you should visit the shop of electronic components [1] and buy in adequate amounts:

circuits ATMEGA8-16PU
solderless field (see fig.)
28-pin socket for chips
10 resistors 330 Ohm, 10 Ohm resistors 1, 5 resistors of 100 Ohm
LEDs according to their own taste, a little seven-segment display
LPT cable (computer-printer or a similar building for the programmer)

(Sooner or later, something will be missing as well. Boxes full of bananas from electronic scrap to rozpájení in this irreplaceable.)

ATMega is programmed via the parallel port

To upload your own program in FLASH memory ATMega may be as follows:

Write on their computer program in C language,
compile it in the GCC,
if you do not already make you programmer,
ATMega connect to the programmer and programmer of the computer's parallel port,
loaded into memory ATMega compiled program using avrdude,
ATMega return to your circuit and run it.

Production of simple programmer STK200

Attention parallel port on the computer has holes. Do not confuse it with the wide version of the serial port, which has pins.

Hardware programmer is a device that is used to connect the microcontroller and PC so that the microcontroller to put the record program. Here we describe how one simple to produce.

There are also more practical version of programmer usbasp that connects to your computer via USB. Its production of which we describe, but because it itself contains a microcontroller without the programmer can not do.

ATmega big advantage is that the program upload is no need for complicated electronics. Programmer for building

known as the STK200 can get by with:

parallel port on a normal PC,
4 resistors (three 330 ohms, a 100 Ohm)
The standard base on chips (with 28 holes) and
standard parallel printer cable.

ATMega during the recording of the program requires a power supply + 5V Most other guides it solves by an additional stabilized AC power source. Current consumption is very low at Atmega8. If the microcontroller is connected to additional stress, we can use the power directly from the parallel cable. Just find a pin, which is permanently turned on during programming logic 1, ie + 5 V. In this case, the pin 14. If the program fails, firstly with a voltmeter to check whether the pin is permanently voltage of 5 V and possibly programmed chip can power an external source. Then make sure that the country must be common to both the source and the computer.

Particular embodiments leave to the reader. If you are working on a solderless field is probably the most practical to the relevant cables directly solder the resistors and colored wires and secure them against short circuits. Slot with resistors can be built into the connector housing, creating a very compact programmer without cable. Another possibility is to incorporate the base into the wall of a cardboard box with tiskárenským connector, as illustrated by images.

Compiling a program in assembler flashing LEDs

This part needs to replenish.

Wikiknihám can help by making it properly you add .

Assembler is a low level language in which to write the specific steps that the processor has to perform. It is suitable where it is necessary to write very efficient and fast code. For normal programming recommended to use the C language.

Listing language program assembler

.NOLIST .INCLUDE "/usr/avr/inc1/m8def.inc" ; odkaz na inc soubor .LIST .DEF tmp = R16 ; tmp - registr pro docasna data rjmp RESET ; provede nezbytne nastaveni mikroprocesoru - inicializace RESET: LDI tmp , LOW (RAMEND) OUT SPL , tmp LDI tmp , HIGH (RAMEND) OUT SPH , tmp rjmp Start ; nyni spustime vlastni program Start: ; sem nen í dobre psat prikazy , stane se program spatne citelny LDI tmp , $01 ; do tmp nahrajem hodnotu 0 b00000001 - > tzn 1 OUT DDRB , tmp ; a tu pak nahrajem do DDRB - takze urcime vstupy a vystupy sbi PortB , 0 ; rozsviti LED rcall Cekej ; spusti 1 s zpozdeni cbi PortB , 0 ; zhasne LED rcall Cekej ; opet spusti zpozdeni rjmp Start cekej: LDI R19 , 14 ; do registru R19 zapise hodnotu 14 v destkove soustave clr R20 ; vymaze data v registru R20 clr R21 ; vymaze data v registru R21 cekej1: dec R21 ; odecte z registru R21 cislo 1 (zmensi o 1) brne cekej1 ; pokud bude hodnota v R21 rovna 0 , pak program pokracuje na dalsi radek , jinak skoci dec r20 brne cekej1 dec r19 brne cekej1 ret ; navrat z podprogramu

Install Packages

gcc-avr avr-libc Avra

We will compile a program in assembler

Avra blikajici_LED.asm

Translated program will therefore appoint blikajici_LED.hex

Downloading programs ATMega
Install the package avrdude:

sudo apt-get install avrdude

To upload your own program to the MCU exists for Linux and a few other programs, for example. Uisp or ponyprog. Atmel supplies a complete development environment AVRStudio that supports programming through parallel programmer as well. Rather we are here for reasons of space deal.

Connect Atmegu8 a computer programmer using the STK200 and run the command:

sudo avrdude -p m8 -c usbasp -U flash: w: main.hex

It is likely that this command you use frequently. You can create so for him. Alias! Add the following line at the end of your ~ / .bashrc:

avrload () {sudo avrdude -p m8 -c usbasp -U flash: w: $ 1; }

Now simply place the above command run only

avrload main.hex

Switching fuses

Atmega8 also has 16 so-called. Fuses, English fuses. They are 2 bytes of EEPROM memory, referred to as lfuse and hfuse whose value is retained when you turn off the circuit and to determine which pins have what function, what the source of clock signals used Atmega8 like.

The value to which they switch fuses, the program passes avrdude as a parameter. For the calculation of these parameters can be used: http://www.engbedded.com/fusecalc/

Very easy to become that man when programming fuses and confuses either set an unusable source clock, or turn off the option to reset the chip. Atmega8 with incorrect settings then will not work, nor collaborate in programming through ISP. It must then be reset using a second chip and parallel programming, which is documented in the Appendices. Be careful when setting policies.

To facilitate the avrdude Linux

Open the file in a text editor ~ / .bashrc and at its end, copy the following three lines. Applies if you program Atmegu8 with simple programming cable "STK200".

```
avrload () {sudo avrdude -p m8 -c STK200 -U flash: w: $ 1; }
avrfuseXtal () {sudo avrdude -p m8 -c -U STK200 lfuse w: 0xff: m -U hfuse w: 0xd9: m; }
avrfuseIntOsc () {sudo avrdude -p m8 -c -U STK200 lfuse w: 0xc1: m -U hfuse w: 0xd9: m; }
```

These shortcuts course, you can add or modify as needed. If instead you use a cable usbprogramátor which is given in a later chapter, replace the STK200 for usbasp. Furthermore, if you do not use ATmega8, ATmega16 but necessary, replace the parameter for m8 m16. In order avrload note the string $ 1, for which there completed the first parameter that you specify for the command avrload.

After you restart the terminal, or upload edited ~ / .bashrc command source ~ / .bashrc, you'll be able to load a program into the chip mujprogram.hex:

```
avrload mujprogram.hex
```

... And switch it to an external clock crystal:

```
avrfuseXtal
```

or to reset the clock at 1 MHz oscillator built:

```
avrfuseIntOsc
```

Program counting pulses

This part needs to replenish.

Wikiknihám can help by making it properly you add .

Output at seven-segment display

Using the input / output ports

Program for easy switching on and off the pin on PORTB.1 button on PORTB.0. There are also resolved glitches buttons and continuing until after releasing the button.

Reading the value pin

```
#include <avr / io.h>
#define F_CPU 8000000UL
#include <util / delay.h>
int main (void)
 {
   DDRB = 0x06; // controlling the input and output, 1 output, 0-log entry
   PORTB = 0x07; // output register and DDR == 1 then the port is transferred to the output differently based log pullup
1-ON, a pullup log 0-OFF
   do {
     if (! bit_is_set (PINB, 0))
      {
        _delay_us (500);
       if (! bit_is_set (PINB, 0))
        {
     if (bit_is_set (PINB, 1))
      {PORTB & = ~ ((1 << 1));}
       else
            {PORTB | = (1 << 1);}
           while (! bit_is_set (PINB, 0))
     {};
        };
      }
   } While (1);
   return (0); // return 0 = termination right
 }
```

To delay runtime, which in microcontrollers uses quite often, you can use the function _delay_ms () and _delay_us (), whose argument is the time in milliseconds, respectively. microseconds. We recommend that those functions are always called static value; Indeed, if they pass a variable in the program nalinkuje of magnitude 10K libraries for mathematical calculations. For proper timing define the value F_CPU as clock frequency (ending UL).

Using the A / D converters

For a simple analog-to-digital conversion is necessary to involve AVCC pins (20) and AREF (21) to a supply voltage of 5 V. assume that the following code segment measures the voltage at the input ADC0 (pin 23).

```
  ADMUX = 0;  // Initialize the A / D converter channel 0
      ADCSRA | = _BV (ADEN) |  _BV (ADPS1) |  _BV (ADPS2);

      ADCSRA | = _BV (ADSC);  // Start measuring
      while (! (ADCSRA & _BV (ADIF)));  // Awaits completion
      ADCSRA | = _BV (ADIF);
```

A / D converter has 10-bit accuracy, ie the value "0" corresponds to 0 V, the "1023" corresponds to 5 V. The bottom 8

bits are stored in the registry ADCL, the upper two bits into the register ACH. We will add a section of code that this 10-bit value is converted to a 4-digit decimal number in ASCII format and send it over the serial port.

```
   UBRRL = 25;  // Init USART 2400 bps (assuming 1MHz internal clock)
       UCSRB | = _BV (TXEN);
       int VInput;
       VInput ADCL = + 256 * (ACH & 3);  // 10 bits ADC input
       usart_putchar (0x30 + (% VInput 10000) / 1000);
       usart_putchar (0x30 + (VInput% 1000) / 100);
       usart_putchar (0x30 + (VInput% 100) / 10);
       usart_putchar (0x30 + (VInput% 10));
       usart_putchar (0x0A);  // End of line
```

When this function is used

```
void usart_putchar (uint8_t c) / * {{{* /
{
   UDR = c;
   while (bit_is_clear (UCSRA, TXC));
   UCSRA = _BV (TXC);
}
```

Description of communication through the serial port, see the relevant chapter.

Accurate measurement

stabilize the voltage at AVCC (20) and the AREF (21) (condenser and / or a stabilizer LM7805)
slow measurements - reducing the clock frequency ADC

If we have more demands, you can also use an external ADC and read data from it via the SPI. For about 70 CZK can be purchased eg. MCP3202 12-bit dual channel transmitter, which incorporates a sample-and-hold circuit. If you want to use a higher sampling rate can be selected eg. Eight-bit AD9280 which samples with frequencies up to 32 MHz. Data then should temporarily stored in external memory (see [2]).

Further information is provided in the datasheet and the
http://www.avrbeginners.net/architecture/adc/m8_adc_example.html .

Writing and reading internal EEPROM memory

This part needs to replenish.

Wikiknihám can help by making it properly you add .

```
#include <avr / eeprom.h>

eeprom_read_block (data, eeprom_offset, len);
eeprom_write_block (data, eeprom_offset, len);
```

Overclocking to a higher frequency of the crystal

This part needs to replenish.

Wikiknihám can help by making it properly you add .

Using interrupt

This part needs to replenish.

Wikiknihám can help by making it properly you add .

Communication with PC through USART

Atmega8 is equipped with the USART , who cares about communicating over the serial port. To send and receive data from the computer because just a few commands. If your computer does not have a serial port, you can replace a variety of ways using a USB port, which is dedicated to some of the following chapters.

Hardware

Serial port (in this case) will use only three wires: ground, data sending and receiving data. Because the data transfers do not use a clock signal, that would point to the arrival of each new bit, the computer must be precisely determined and chip common baud rate.

These frequency ("baud rate") are standardized. In this example, we assume that using the internal oscillator at 1 MHz ATMega and use a baud rate of 2400 bit / s. If we used ATMega clock crystal frequency of 4 MHz, we would use instead of 9600 bit / s.

Atmega8 uses TTL voltage levels (0 V / 5 V). PC serial port uses a significantly different voltage levels (-12 V / +12 V). The transfer between these levels should be used MAX232, which requires connecting additional four capacitors. So buy MAX232CPE circuit (about 19 CZK) and 4x him electrolytic capacitor 1 uF / 20 V.

ATMega firmware to communicate via USART

```
#include <avr / io.h>

// Get a char
unsigned char uart_getc (void)
{
    / * Wait for data to be received * /
    while (! (UCSRA & (1 << RXC)))
        ;
    / * Get and return received data from buffer * /
    return UDR;
}


// Send char
void uart_putc (unsigned char data)
{
    while (! (UCSRA & (1 << UDREs)))
        ;
    UDR = data;
}


int main (void)
{
    char character;

    / UART initialization ***** ***** /
    UCSRA = 0x00;
    UBRRH = 0x00; // Set the speed to 1 MHz crystal
    UBRRL = 25; // (in case of 8 MHz clocked here Use UBRRL = 0x33;)
    UCSRB = 0x18; // enable the transmission and reception
    UCSRC = 0x86; // data frame: 8 data, 1 stop bit, no parity

    while (1)
    {
        uart_getc character = ();
        uart_putc (character);
    }
}
```

Software on the computer
3 We describe the environment in which they can communicate with ATmega. Which one is best depends on the application.

Linux command line (entry in the / dev / ttyS1)
program in Python
program in C

Linux Command Line
Receiving data from the serial port devices, accessible as / dev / ttyS0 or / dev / ttyS1. If you're lucky, it will directly read data:

cat / dev / ttyS0

Received data was flowing to standard output, as is customary.

If it does not, read from port probably requires adding a few parameters:

```
stty -F / dev / ttyS0 clocal cread -crtscts CS8 -cstopb hup -parenb parodd -brkint -icrnl ignbrk
  -igncr ignpar imaxbel -inlcr inpck -istrip -iuclc -ixany ixoff -ixon bs0 CR0 FF0 nl0 -ocrnl
  -ofdel -ofill -olcuc -onlcr -onlret onocr -opost Tab0 vt0 -crterase crtkill -ctlecho -echo
  -echok -echonl -echoprt -icanon -iexten -isig -noflsh -tostop -xcase time 5 min 1 2400
```

Todo: data

Program in Python

This part needs to replenish.

Wikiknihám can help by making it properly you add .

For receiving data in Python, there is an easy to use library USPP, which will probably require installation (sudo apt-get install python-USPP).

Advanced communications solutions via a serial port
Firmware providing complete communication using USART interrupt.

Chip type: ATmega16
Clock frequency: 12MHz
jan.kalenda [at] gmail.com

```
#include <avr / io.h>
#include <avr / interrupt.h>
```

```
#include <string.h>

#define rdataSize 100

// USART
void USART_Init (unsigned int divisor);
void USART_Transmit (char * data);
void usart_busy_wait ();

Tdata char = 0, * = transmitted_data & Tdata;
rdata char [rdataSize] * = received_data rdata;


#define UBRR 77

int main (void)
{
 USART_Init (UBRR);
 sei ();

 while (1)
 {
              usart_busy_wait ();
  USART_Transmit (rdata);
 }

 return 0;
}

void USART_Init (unsigned int divisor)
{
 // Set the port as the TxD output.
 DDRD | = 0x02;

 // Set baud rate
 UBRRH = (unsigned char) (divisor >> 8);
 UBRRL = (unsigned char) divisor;

 // Set frame format: 8data, 1 stop bit.
 UCSRC = (1 << Ursel) | (3 << UCSZ0);
 // Enable Receiver and Transmitter
 UCSRB | = (1 << RXEN) | (1 << TXEN);
 UCSRB | = (1 << RXCIE);

}

usart_busy_wait void ()
{
 / * Wait for empty transmit buffer * /
 while (transmitted_data! = NULL);
 while (! (UCSRA & (1 << UDREs)));
}


void USART_Transmit (char * data)
{
 transmitted_data = data;
 if (* transmitted_data)
 {
  // Enable Data Register empty interupt
```

```
   UCSRB | = (1 << strikes);
  }

 }

 // Data transfer using data register empty interrupt.
 ISR (USART_UDRE_vect)
 {
  // Put data into buffer, sends the data.
  if (* transmitted_data)
  {
   UDR = * transmitted_data;
   transmitted_data ++;
  }

  else
  {
   // Disable Data Register empty interupt
   UCSRB & = ~ (_BV (strikes));
  }
 }
 // Receive data using USART Receive Complete interupt
 ISR (USART_RXC_vect)
 {
  if (received_data! = & rdata [rdataSize])
  {
     * Received_data = UDR;
     // UDR = * received_data;
     received_data ++;
  }
  else
  {
   received_data = rdata;
  }
 }
```

Recording to SD memory card
http://www.mikrozone.sk/pluginy/content/content.php?content.43

Communication with PC via USB
USB has over the serial port baud rate and higher is supported even new computers. Also includes power conductor (5 V, 500 mA). A substantial disadvantage is the complexity of communication by the device via USB lead. There are several ways to connect microcontroller with computer:

- implement USB protocol to ATMega software - for there are a number of independent projects
- http://www.obdev.at/products/avrusb/index.html - clear and relatively easy to use functionality to communicate via USB, including many examples, support different microcontrollers and different frequencies for commercial purposes charged
- http://www.cesko.host.sk/IgorPlugUSB/IgorPlug-USB%20(AVR).htm - popular (and one of the first) projects for Atmel USB, RS232-USB
- http://smrz.chrudim.cz - a good description of communication via USB, but specific instructions require a lot of their own effort
- http://www.volny.cz/elecon/cz/usb/usb_m.html

http://www.marktmarshall.com/doku.php?id=projects:avr-hid

use a microcontroller that has a built-in USB interface (Atmega8 it is not, these microcontrollers are considerably more expensive)
use additional circuit that ensures transfer between USB and RS232 (serial RS232 interface already supports Atmega8)
FT232BL (http://www.datasheetsite.com/datasheet/FT232BL) costs about 130 CZK and can achieve communication with him about 2 Mbit / s, which is significantly more than can the software USB driver. FT232BL series of instructions for them on the Internet.

The author has a good experience in using the first option. For their experiments zvolis USB driver from Objective Development. For commercial use are free under the GPL. To try it out, download and unzip their source codes in zip format and examine examples.

For easy communication with computer is suitable eg HID class device-DATA. Circuit proved to be a zener diode (also referred to in that archive). Further description of communication via USB is beyond the scope of this Wiki.

Construction USB Programmer
In one of the opening chapters described the construction of a simple programmer STK200, utilizing parallel port. Because some computers already have a parallel port, there are a series of instructions for building a programmer which connects via USB. Some of them have advanced features such as parallel programming, support for a wide array chip debugging, etc.

For most purposes, but will do a simple programmer usbasp by page http://www.fischl.de/usbasp/ . Enable continuous programming several microcontrollers from Atmel (including Atmegy8 and Atmegy16). Scheme and the program can be found in the archive to the author, the programmer may be a few tens of minutes to deploy solderless field.

The author had trouble with the newer version of avrasp 2009. [http://www.fischl.de/usbasp/usbasp.2007-10-23.tar.gz

atmega8
Crystal 12.000 MHz
ICE
5 x 330 ohm, 33 ohm 2x 2x 100 ohm, 1 1k
B USB connector or cut end of the USB cable that can connect to computers

Atmegu8 need to switch to an external clock crystal and programmed. So if you have no parallel port, or borrowed programmer, you need to contact someone to record the program.

Appendices
Short glossary

Atmega8 - middle-class microcontroller manufactured by Atmel, belongs to the family of AVR
Atmel - one of many companies producing microcontrollers, among other things produces Atmegu8, which deals with this book
AVR - family processors from Atmel product line. These include Atmega8.
AVRStudio - rich development environment for Atmel MCUs, see [3]
avrdude: program to record the program to AVR MCU series
EEPROM rewritable electronic memory, similar to flash (for Atmegy8 512 B used to store data)
FLASH: electronically rewritable memory, similar to flash (for Atmegy8 8 KB to store the program)
FUSE: here is one of a few bits in the permanent memory of the microcontroller, which determines its behavior (Czech also "fuse")
ISP - In System Programming: serial programming via 4 data lines, usually without having to remove the device from a

microcontroller that controls
JTAG: debugging interface program recorded in the microcontroller (also ICE)
LPT: parallel port
microcontroller programmable integrated circuit capable of autonomous action without further complicated electronics, similar to its logical structure greatly diminished computer
Programmer: here refers to a circuit through which the microcontroller to upload data and settings from your computer
RS232 serial port
Serial port: port older, even sometimes occurring in desktops. The data is transmitted one conductor therein and a second back (3 conductors, for some purposes even more). Uses the voltage level of +/- 12 V, which do not normally used voltage in digital circuits.
STK200: very simple programmer, consisting of cable and 4 resistors, connects to the parallel port
uisp: program to record the program to the AVR microcontroller series, similar avrdude
USART - Universal Synchronous or Asynchronous Receiver and Transmitter: part of the microcontroller, providing hardware via serial port
USB - Universal Serial Bus: a very popular port for existing desktops to which you can also connect a single-chip computers and using
USB 1.1: The older USB standard, which operates at a transmission speed of 1.5 Mbit / s or 12 Mbit / s. Speed ??of 1.5 Mbit / s can communicate with the help of a software driver in the microcontroller; higher speeds require a dedicated circuit.
usbasp: relatively simple programmer, connects to USB
Watchdog timer: independent circuit, preventing lockups run microcontroller. If you cease to be periodically "feed", he regards it as a collapse of the program and reset the circuit.

High parallel programming
Motivation
To upload the program to Atmegy8 and setting its policies have been using serial programming. (Byte in it serially transmitted over wires 3, although the computer uses the parallel port.) Also known as In System Programming, because they often can be done without Atmega8 took out of the circuit.

In some cases it is necessary to use a complicated high voltage parallel programming:

They are set incorrectly fuses, which can disable serial programming

can be reset fuse RSTDSBL, which causes the pin PC6 ceases to serve as a reset
or are confused programmed fuses source clock (clock while at the ISP required)

We want to read or write data in the EEPROM
We want to erase the data on the chip, which is locked against writing (as set lock bits)

In the following instructions we will deal first and most common case where we have mistakenly set the fuses and the chip has not overridden by using the ISP. The chip mark as ill patient. Using the second atmegy8 chip reset so that it could re-programmed using the ISP.

Required materials
The parallel programming can not use the simple computer's parallel port, because it has enough pins. Instead, we'll get

second, functional Atmegu8. We'll refer to as a doctor. (The fuse left in the default state, ie. Internal clock at 1 MHz).
stabilized power supply 11.5 to 12.5 V
1 NPN transistor, one PNP transistor
4 resistors of 10 Ohm
standard equipment previously used, ie. solder field, serial programmer, supply voltage of 5 V and a pile of wires

Program upload doctor
The doctor will record the following recipe: Programujeme_jedno?ipy / VnParProg_hex_code :

sudo avrdude -p m8 -c STK200 -U flash: w: atmega8-fuse-repair.hex

If you want to modify the program or read, is available here: Programujeme_jedno?ipy / VnParProg_c_code . In this case, compile the classical orders and upload it as described above.

```
avr-gcc -g -Wall -O1 -mmcu = atmega8 -c atmega8-fuse-repair.c -o atmega8-fuse-repair.o
avr-gcc -g -Wall -O1 -mmcu = atmega8 atmega8-fuse-repair.o -o atmega8-fuse-repair.elf
AVR-objcopy j j .text .data O iHex atmega8-fuse-repair.elf atmega8-fuse-repair.hex

sudo avrdude -p m8 -c STK200 -U flash: w: atmega8-fuse-repair.hex
```

The repair procedure fuses

Both Atmegy8 placed on the solder field close to each other. Most of the pins connect directly to PB0 PB0 etc. This will create PB0 - PB6, PC0 - PC2, PD1 - PD7, a total of 17 straight jumpers that are on the diagram drawn merged in a thick line:

PC3 connect the output to the input of the switch +12 V output switches connect via a 10 Ohm resistor to pin PC6 patient. The diagram is a draft with transistors, but because the 12 V signal is switched on for programming, can be transistor circuits replacing manual switch and a single resistor.

Both chips, the doctor and the patient, connect according to the above scheme. We recommend that you then proceed as follows:

RESET Dr. connect to earth
Connect the stabilized 12 V and 5 V power supply
(If we simplified diagram of removing transistors connect 12 V to reset pin of the patient)
Disconnect the RESET and monitor light emitting diode should blink for about a second

Now, the patient should be cured. Power supply fails, you take it and try to program a conventional, serial interface.

Direct entry to the parallel port in Linux

It is not related directly to ATmega, but it might come in handy. Lptout following program (written in C for Linux) is set to PC parallel port number specified as the first parameter. On Windows (XP) can write to the parallel port as well, but it requires the installation of additional drivers. Writing to LPT only works when it is running with root privileges.

```
/ * Lptout - sets given value on parallel port * /
 #include <stdio.h>
 #include <stdlib.h>
 #include <unistd.h>
 #include <sys / io.h>
 #define base 0x378 / * printer port address * /
 int main (int argc, char ** argv)
 {
   int value;
   if (ioperm (base, 1, 1)) {fprintf (stderr, "Could not get permission for the port 0x% x. n", base) exit (1);}
   if (argc <2) {fprintf (stderr, "Please specify the value to be set. n"); exit (2);}
   if (sscanf (argv [1], "% i", & value) == 1) {outb ((unsigned char) (value), 0x378);}
```

```
      else {fprintf (stderr, "1st parameter must be an integer Between 0 - 255.  n", base) exit (3);}
    return 0;
}
```

Compile and an example to set pins nast?ída?ku:

```
g ++ -o lptout.c lptout
sudo ./lptout $ ((1 * 1 + 0 * 2 + 1 * 4 + 0 * 8 + 1 * 16 + 0 * 32 + 1 * 64 + 0 * 128))
```

Be careful when working with parallel port! Computer circuits are sensitive to surges and static electricity. For parallel ports, the risk of damage in the event of a short circuit.

Useful links

http://robotika.cz/guide/blink/cs - detailed instructions on how to load a program ATMega
http://pramode.net/articles/lfy/atmega8/pramode.html - Communication with PC via a serial port

categories :

Maintenance: Amend section
Tutorials