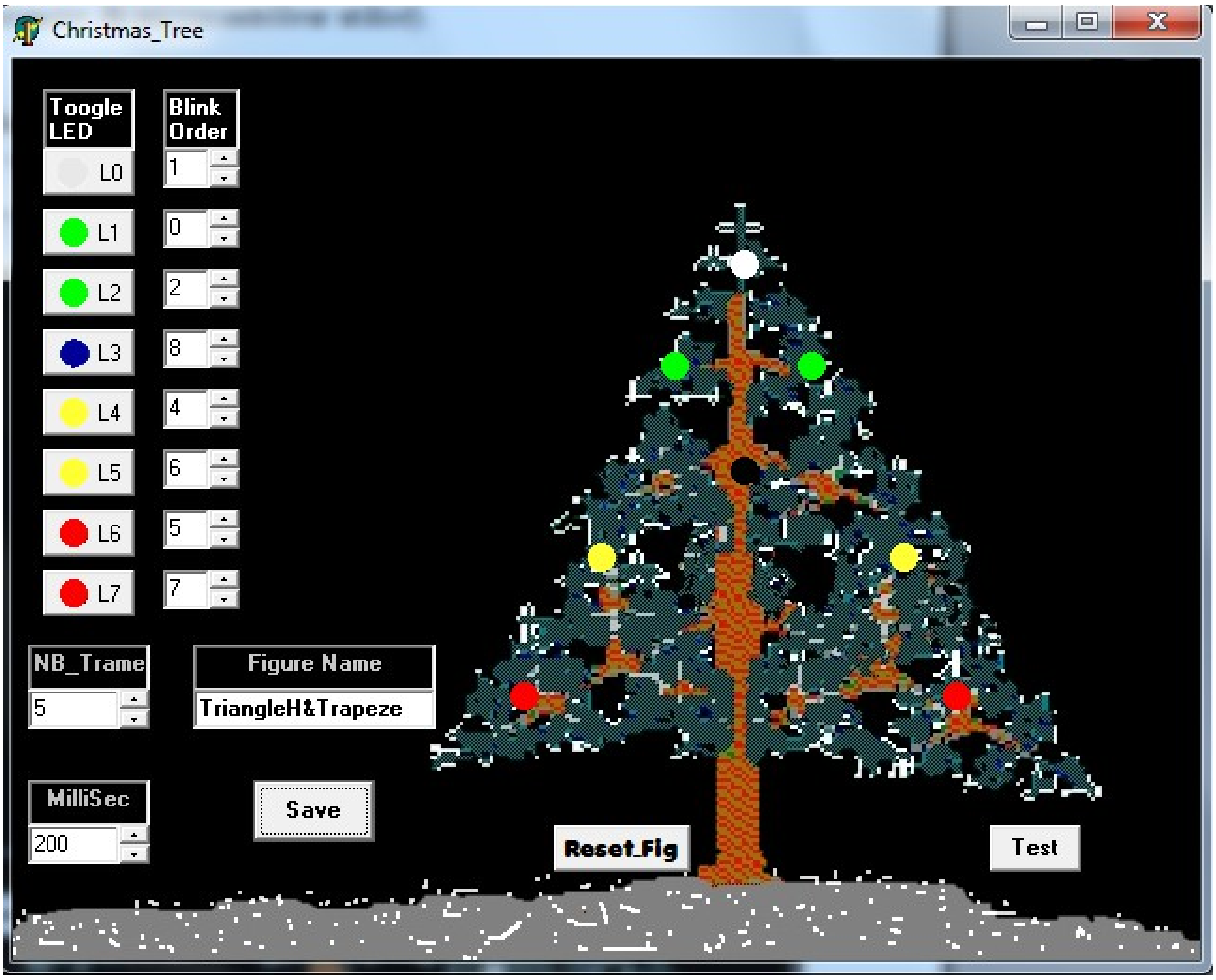
TREE8LED Version 1.0 27/02/2015 (Delphi 7) (Sorry for my English)

The goal of the program TREE8LED is to create animated sequence LED with mouse or keyboard
Each animation sequence can be saved as a TXT file that will fit in a C program for Arduino or AVR Studio for the range of microcontrollers ATtiny13 / 45 / 85.
A Printed Circuit shaped mini Christmas Tree made up of eight LED four resistors
and a microcontroller (SMD format in my case) was specifically designed to display multiple animation sequence (limited to the memory of the microcontroller used).

IMPORTANT:
Put all Blink Order box 8 (with Reset_Fig button) before starting a figure or to reset a figure, if the figures you create will be wrong.

Time in millisec must be adapted because microcontroller as ATtiny13 and Attiny85 have an internal oscillator which operates at a different frequency.
At the end of the tutorial, you will find information for Oscillator speeds and programming FusesBit.



Toogle LED	Blink Order
L0	1
L1	0
L2	2
L3	8
L4	4
L5	6
L6	5
L7	7

NB_Trame: 5
Figure Name: TriangleH&Trapeze
MilliSec: 200

Screen shoot

BUTTONS:

Reset_Fig
this button resets all digital boxes zero (8)
NB_Trame
This number indicates the number of times that the figure will be displayed.

Millisec
This figure shows the ignition time by LED Millisecond

Figure Name
It was under this name that the figure will be saved with the extension .txt (TriangleH & Trapeze.txt on the screenshot)

Toogle LED (Switch LED)
This button displays the status of the eight LEDs and acts as a switch that is used to turn on or off the **LED buttons give the LED position on the tree which gives an idea of what will figure on the electronic circuit** but are not used for programming the order of LED displayed by the microcontroller and **not used in the microcontroller flash memory.**
L0 represents the LED 0, L1 LED 1 ... . L7 LED 7

Blink Order
This arrow button (increases / decreases) is important because it is he who manages the firing order and extinguishing LED (0 for the first and for the last 7, 8 indicates an equal footing to the time in milli second),
5 indicates that the figure is repeated 5 times
after we see that L1 lights first then L0 and L2 and then there was a pause of 200mS because L3 does not light then illuminates L4 and L5 and L6 and L7).

When all these settings are correct you can click on the **Save button** and the file is created in the current directory.

Here is the generated file to be integrated into the SomeOne procedure for Arduino or April Studio:
SomeOne (5,1,0,2,8,4,6,5,7,200) // TriangleH & Trapeze

the displayed speed of 200 milli seconds for the ATtiny13's, if you use 85 or Attiny45
multiply the speed by about 8 or 10.

Here is the table for example ATtiny13 and Attiny85:

moves in one ATtiny13:

```
  SomeOne (8,0,1,2,3,4,6,5,7,150); // One at a very slow
  SomeOne (8,0,1,2,3,4,6,5,7,65); // One by one medium
  SomeOne (15,0,1,2,3,4,6,5,7,25); // One by one  fast
  SomeOne (90,0,1,2,3,4,5,6,7,1); // flash
  SomeOne (8,1,2,6,4,3,8,8,0,35); // trapeze & L4 Fast
  SomeOne (10,0,1,2,8,4,5,6,7,40); // big TriangleparL1
```

Moves in one or Attiny45 85:

```
  SomeOne (8,0,1,2,3,4,6,5,7,1200); // One at a very slow (150 on ATtiny13)
  SomeOne (8,0,1,2,3,4,6,5,7,600); // One by one medium
  SomeOne (8,0,1,2,3,4,6,5,7,300); // One by one fast
  SomeOne (80,0,1,2,3,4,6,5,7,0); // flash
  SomeOne (8,1,2,6,4,3,8,8,0,600); // trapeze & L4 Fast
  SomeOne (10,0,1,2,8,4,5,6,7,500); //  big TriangleparL1
```

Here is the complete program for ATtiny13, only 85 for the display to change speeds are as above:

```
*
 * For TreeOneByOne.ino    ATtiny 13
 *
 * Created: 5/29/2015 2:42:12 p.m. Arduino 1.06
 * Author: Mic-Josi
 * Tiny Tree for Rev. 1.1 and 1.7 PCB
 * Tiny Tree Rev. 1.1 and 1.7
 * Internal oscillator selection Configuration 4.8 MHz and dividing by 8 to 1.8 Volts power failure
 * Setting 4.8 MHz oscillator and internal division by 8 selected brown-out detection level at 1.8
Volts
 * Fusebits: Low Hight = 0x69 = 0xFD
 * /


/ *
These are for using #define inused with Amtel Studio 6 only
These olefins are for use only with Amtel Studio 6
#define __DELAY_BACKWARD_COMPATIBLE__
#define F_CPU 8000000UL
#include <April / io.h>
#include <Util / delay.h>
#define Byte uint8_t
* /

// Arduino defines
#include <April / pgmspace.h>
#define LED_COUNT 9
#define DDR_BYTE 0
#define PORT_BYTE 1
```

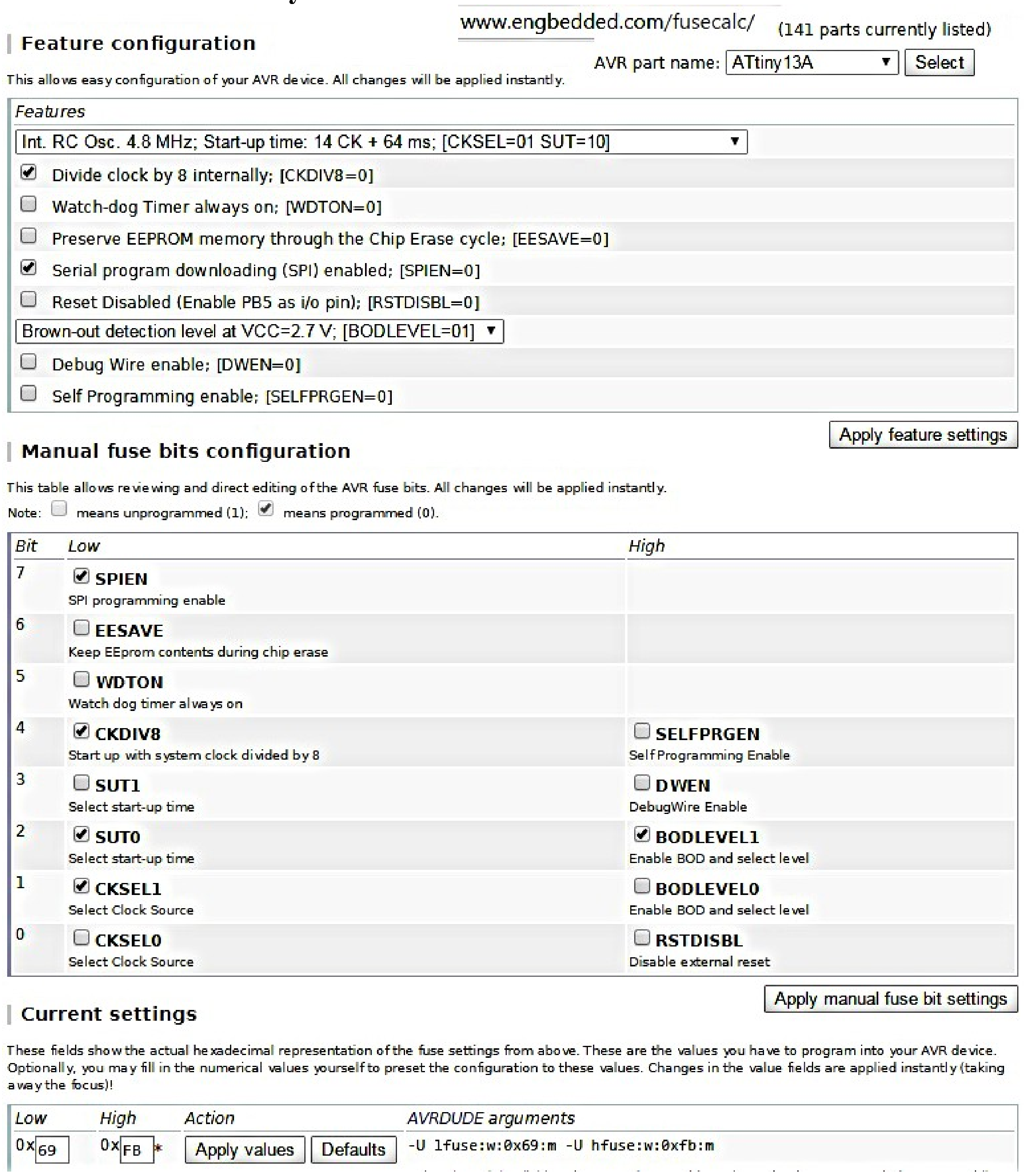
```
const byte matrix[LED_COUNT][2] PROGMEM = {

        // DDR_BYTE       PORT_BYTE
        {0b00010001       , 0b00010000},//L0-PB4            0
        {0b00010001       , 0b00000001},//L1-PB4         "     "
        {0b00001001       , 0b00001000},//L2-PB3        " 1    2 "  **     Pin-Tree     **
        {0b00001001       , 0b00000001},//L3-PB3       "      3     "        L0+ L1- Pin 3
        {0b00000101       , 0b00000100},//L4-PB2     " 4           6 "     L2+ L3- Pin 2
        {0b00000101       , 0b00000001},//L5-PB2    "                  "     L4+ L5- Pin 7
        {0b00000011       , 0b00000010},//L6-PB1    " 5               7  " L6+ L7- Pin 6
        {0b00000011       , 0b00000001},//L7-PB1   ** +- Commun PB0 Pin 5 **

};


// ************************************************** *******
void TurnOn (byte led) // PBx selection
{
  DDRB pgm_read_byte = (& (matrix [led] [DDR_BYTE]));
  Pgm_read_byte PORTB = (& (matrix [led] [PORT_BYTE]));
}
// ************************************************** *******


// ************************************************** *******
// Frame- number of times in the figure, L1 to L8 = LED status, spd = time on an LED in mili-
second
```

void SomeOne (frame byte, byte l1, l2 byte, byte l3, l4 byte, byte l5, l6 byte, byte l7, l8 byte, byte spd) // Some LEDs ON one by one spd time

```
{
case 1:
figure [s] = l2;
break;

case 2:
figure [s] = l3;
break;

case 3:
figure [s] = l4;
break;

Box 4:
figure [s] = l5;
break;
Box 5:
figure [s] = l6;
break;

Box 6:
figure [s] = l7;
break;

Box 7:
figure [s] = l8;
break;

Box 8:
figure [s] = l9;
break;
}


}
for (the byte = 0; l <LED_COUNT; s ++)
{
= lights figure [s];
TurnOn (lights);
delay (spd);
}

light ++;
}
delay (50);

}

 // ************************************************* *******

void setup ()
{
}


// ************************************************* *******
// ================================================= =======
void loop ()
{
 / * 0-7 ignition location of the LED, LED off 8
   0 to 7 of the LED lighting rental, 8 LED off * /

  SomeOne (8,0,1,2,3,4,6,5,7,150); // One at a very slow
  SomeOne (8,0,1,2,3,4,6,5,7,65); // One by one medium
  SomeOne (15,0,1,2,3,4,6,5,7,25); // By a rapid
  SomeOne (90,0,1,2,3,4,5,6,7,1); // Flash
  SomeOne (8,0,1,2,8,8,8,8,8,70); // little slow triangle top
  SomeOne (8,1,2,3,8,1,2,3,8,49); // small inverted triangle top lent0
  SomeOne (8,5,4,1,0,0,2,6,7,42); // Large slow triangle
  SomeOne (8,8,5,4,1,3,2,6,7,46); // Quick largest M
  SomeOne (8,1,2,6,4,3,8,8,0,35); // Trapeze & L4 Fast
  SomeOne (10,0,1,2,8,4,5,6,7,40); // GrandTriangleparL1
    // SomeOne (8,0,2,1,8,4,6,5,7,150); // 02184657
}

// =================================================== =======
```

below the screenshot for FusesBit these microcontrollers:
Here is a copy of configuration fuses screen for ATtiny Attiny45 13 and 13A and 85
the screenshot is that of AVR configuration tool here is the link:
http://www.engbedded.com/fusecalc/

FusesBIT for the Attiny13 et 13A :



www.engbedded.com/fusecalc/ (141 parts currently listed)

AVR part name: ATtiny13A ▾ Select

Feature configuration

This allows easy configuration of your AVR device. All changes will be applied instantly.

Features

Int. RC Osc. 4.8 MHz; Start-up time: 14 CK + 64 ms; [CKSEL=01 SUT=10] ▾

- [✓] Divide clock by 8 internally; [CKDIV8=0]
- [ ] Watch-dog Timer always on; [WDTON=0]
- [ ] Preserve EEPROM memory through the Chip Erase cycle; [EESAVE=0]
- [✓] Serial program downloading (SPI) enabled; [SPIEN=0]
- [ ] Reset Disabled (Enable PB5 as i/o pin); [RSTDISBL=0]

Brown-out detection level at VCC=2.7 V; [BODLEVEL=01] ▾

- [ ] Debug Wire enable; [DWEN=0]
- [ ] Self Programming enable; [SELFPRGEN=0]

Apply feature settings

Manual fuse bits configuration

This table allows reviewing and direct editing of the AVR fuse bits. All changes will be applied instantly.

Note: [ ] means unprogrammed (1); [✓] means programmed (0).

Bit	*Low*	*High*
7	[✓] **SPIEN** SPI programming enable	
6	[ ] **EESAVE** Keep EEprom contents during chip erase	
5	[ ] **WDTON** Watch dog timer always on	
4	[✓] **CKDIV8** Start up with system clock divided by 8	[ ] **SELFPRGEN** Self Programming Enable
3	[ ] **SUT1** Select start-up time	[ ] **DWEN** DebugWire Enable
2	[✓] **SUT0** Select start-up time	[✓] **BODLEVEL1** Enable BOD and select level
1	[✓] **CKSEL1** Select Clock Source	[ ] **BODLEVEL0** Enable BOD and select level
0	[ ] **CKSEL0** Select Clock Source	[ ] **RSTDISBL** Disable external reset

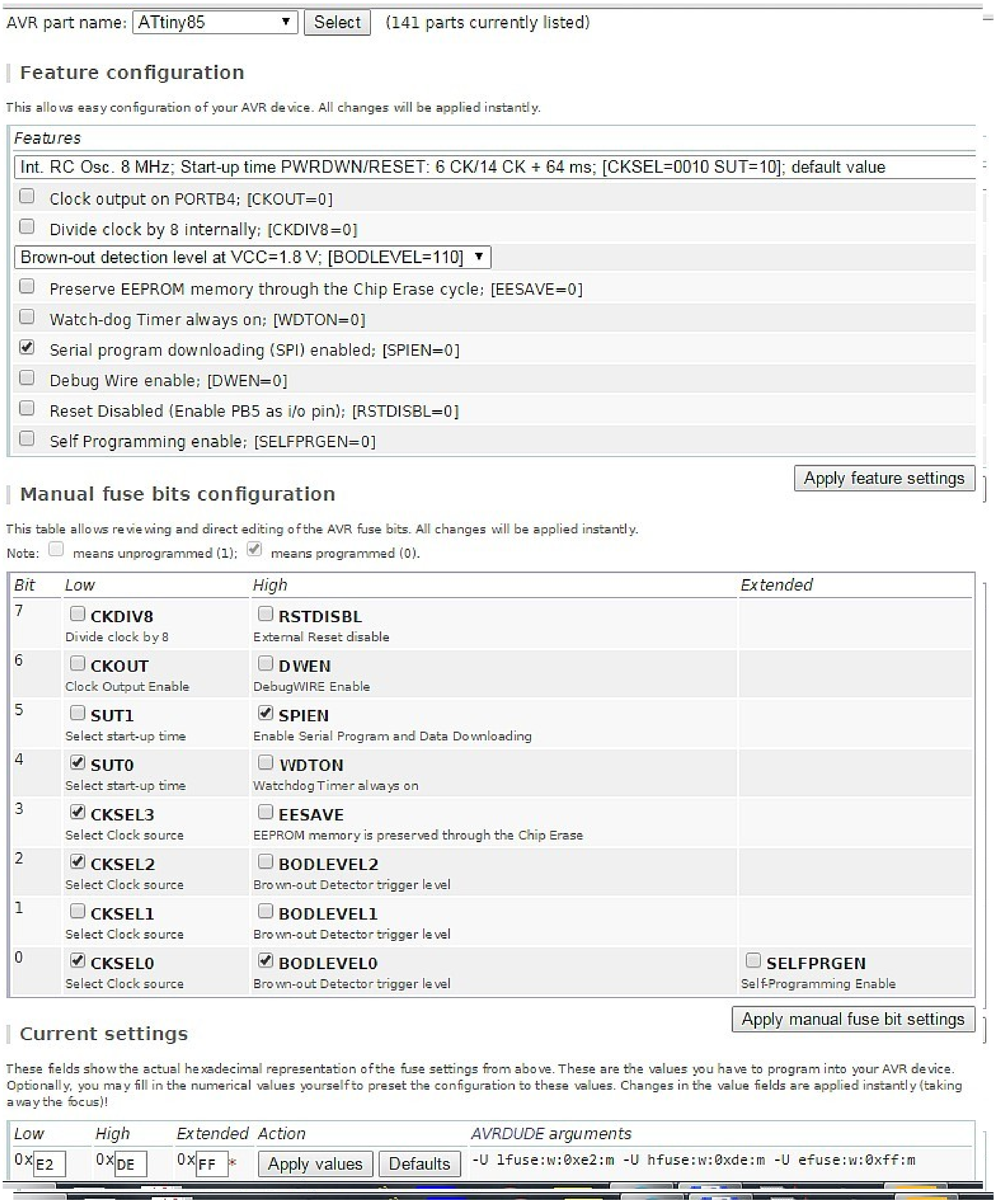
Apply manual fuse bit settings

Current settings

These fields show the actual hexadecimal representation of the fuse settings from above. These are the values you have to program into your AVR device. Optionally, you may fill in the numerical values yourself to preset the configuration to these values. Changes in the value fields are applied instantly (taking away the focus)!

Low	*High*	*Action*	*AVRDUDE arguments*
0x 69	0x FB *	Apply values / Defaults	`-U lfuse:w:0x69:m -U hfuse:w:0xfb:m`

FusesBIT for the Attiny 45 or 85 :



AVR part name: ATtiny85 ▾ Select (141 parts currently listed)

Feature configuration

This allows easy configuration of your AVR device. All changes will be applied instantly.

Features

Int. RC Osc. 8 MHz; Start-up time PWRDWN/RESET: 6 CK/14 CK + 64 ms; [CKSEL=0010 SUT=10]; default value

- [ ] Clock output on PORTB4; [CKOUT=0]
- [ ] Divide clock by 8 internally; [CKDIV8=0]

Brown-out detection level at VCC=1.8 V; [BODLEVEL=110] ▾

- [ ] Preserve EEPROM memory through the Chip Erase cycle; [EESAVE=0]
- [ ] Watch-dog Timer always on; [WDTON=0]
- [✓] Serial program downloading (SPI) enabled; [SPIEN=0]
- [ ] Debug Wire enable; [DWEN=0]
- [ ] Reset Disabled (Enable PB5 as i/o pin); [RSTDISBL=0]
- [ ] Self Programming enable; [SELFPRGEN=0]

Apply feature settings

Manual fuse bits configuration

This table allows reviewing and direct editing of the AVR fuse bits. All changes will be applied instantly.
Note: [ ] means unprogrammed (1); [✓] means programmed (0).

Bit	*Low*	*High*	*Extended*
7	[ ] **CKDIV8** Divide clock by 8	[ ] **RSTDISBL** External Reset disable	
6	[ ] **CKOUT** Clock Output Enable	[ ] **DWEN** DebugWIRE Enable	
5	[ ] **SUT1** Select start-up time	[✓] **SPIEN** Enable Serial Program and Data Downloading	
4	[✓] **SUT0** Select start-up time	[ ] **WDTON** Watchdog Timer always on	
3	[✓] **CKSEL3** Select Clock source	[ ] **EESAVE** EEPROM memory is preserved through the Chip Erase	
2	[✓] **CKSEL2** Select Clock source	[ ] **BODLEVEL2** Brown-out Detector trigger level	
1	[ ] **CKSEL1** Select Clock source	[ ] **BODLEVEL1** Brown-out Detector trigger level	
0	[✓] **CKSEL0** Select Clock source	[✓] **BODLEVEL0** Brown-out Detector trigger level	[ ] **SELFPRGEN** Self-Programming Enable

Apply manual fuse bit settings

Current settings

These fields show the actual hexadecimal representation of the fuse settings from above. These are the values you have to program into your AVR device. Optionally, you may fill in the numerical values yourself to preset the configuration to these values. Changes in the value fields are applied instantly (taking away the focus)!

Low	*High*	*Extended*	*Action*	*AVRDUDE arguments*
0x E2	0x DE	0x FF *	Apply values / Defaults	`-U lfuse:w:0xe2:m -U hfuse:w:0xde:m -U efuse:w:0xff:m`

Here is the link for downloads:
Delphi files KiCad, MPLAB, video and AMTEL STUDIO
http://1drv.ms/1OwZtzi
KiCad software:
http://iut-tice.ujf-grenoble.fr/kicad/
Additional libraries for KiCad: http://www.kicadlib.org/