

# Manual: Programming Firebird Using Esterel

by

**Prof. Kavi Arya(kavi@cse)**  
**Y Shashidhar(shashi@cse)**  
**B Election Reddy(electionreddy@cse)**

ERTS Lab  
Department of Computer Science and Engineering  
Indian Institute of Technology, Bombay  
Mumbai

# Contents

<b>1</b>	<b>Instructions</b>	<b>3</b>
<b>2</b>	<b>Signals for Sensors of Firebird</b>	<b>4</b>
2.1	Buzzer . . . . .	4
2.2	Bump Switches . . . . .	4
2.3	Whiteline Sensors . . . . .	5
2.4	Infrared Range Sensors . . . . .	5
2.5	Battery Voltage . . . . .	5
2.6	Stepper Motors . . . . .	5
2.7	Shaft Encoders . . . . .	6
2.8	LCD . . . . .	6
2.9	tick . . . . .	7
2.10	Sample Program . . . . .	7
<b>3</b>	<b>Kontroller Lab settings</b>	<b>9</b>
3.1	Configure Project Settings . . . . .	9
3.1.1	Common tab settings . . . . .	9
3.1.2	Compiler tab settings . . . . .	10
3.1.3	Linker tab settings . . . . .	11
3.1.4	Assembler tab settings . . . . .	12
3.1.5	Make tab settings . . . . .	13
3.2	Configure Programmer Settings . . . . .	14
3.2.1	Choose Programmer tab settings . . . . .	14
3.2.2	UISP-General tab settings . . . . .	15
3.2.3	UISP-STK500 tab settings . . . . .	16
3.2.4	AVR DUDE tab settings . . . . .	17

# 1 Instructions

1. Download and install *Columbia Esterel Compiler – CEC* from course homepage. This is also available at

- <http://www.cse.iitb.ac.in/~cs684/esterel/cec-0.4.tar.gz>
- <http://www1.cs.columbia.edu/~sedwards/cec>

Installation instructions are given in README file.

2. Download the file *Firebird\_Esterel.tar.gz* from course homepage. This is also available at [http://www.cse.iitb.ac.in/~cs684/esterel/Firebird\\_Esterel.tar.gz](http://www.cse.iitb.ac.in/~cs684/esterel/Firebird_Esterel.tar.gz)
3. Untar it using the command `tar -xvf Firebird_Esterel.tar.gz` . This creates a directory *Firebird\_Esterel* .
4. Create your program, say *prog.strl* , in that directory.
5. *prog.strl* should use only the signals specified in the section 2 to work on firebird. No need to use these for local signals
6. Use the command `firebird_gen prog.strl` . This generates the file *prog.c* .
7. Download and install *Kontrollerlab*. You can download this at <http://sourceforge.net/projects/kontrollerlab> . Ubuntu users can get it through Synaptic Manager.
8. To compile your projects in *Kontrollerlab*, you need to have the following installed in your system
  - **avr-binutils** Binary utilities supporting Atmel’s AVR targets
  - **avr-gcc** The GNU C compiler (cross compiler for avr)
  - **avr-libc** Provides a subset of the standard C library for Atmel AVR 8-bit RISC microcontrollers
  - **avrdude** Software for programming Atmel AVR microcontrollers
9. Open *Kontrollerlab*. You can do this by typing `kontrollerlab` in terminal.
10. To create a new project, go to *Project* menu and choose *New project* option.
11. To add your *prog.c* to this project, open the file *prog.c*
12. Before going for compilation configure your project and programmer as per the instructions given in section 3.
13. To compile the project go to *Project* menu and choose *Build project* option.
14. Successful compilation creates a *hex* file of specified name.
15. Now connect the ISP cable to Firebird.
16. To load the *hex* file on to Firebird, go to *Project* menu and choose *upload* option.

## 2 Signals for Sensors of Firebird

This section details the various kinds of signals modeled for the sensors of firebird.

### 2.1 Buzzer

Firebird has a buzzer, when activated produces alarm. This is modeled as an output signal. This is a pure signal and can take values of on or off. The corresponding signals for those values are

- **BUZZER\_ON** Activate the buzzer for 10 msec.
- **BUZZER\_OFF** Shut off the buzzer

### 2.2 Bump Switches

Firebird has 5 bump switches numbered from 1 to 5. This is modeled as an input signal. This is also a pure signal. A separate signal is used for each bumpswitch.

- **BUMPSWITCH\_1** Input signal for bumpswitch1
- **BUMPSWITCH\_2** Input signal for bumpswitch2
- **BUMPSWITCH\_3** Input signal for bumpswitch3
- **BUMPSWITCH\_4** Input signal for bumpswitch4
- **BUMPSWITCH\_5** Input signal for bumpswitch5

Sometimes we may look for any of the bumpswitches to be pressed. In that case instead of using all the above 5 signals, one more input signal is added for simplicity.

- **BUMPSWITCH\_ANY** Input signal for checking any of the bumpswitches

This signal at the input is activated when any of the bumpswitch is pressed.(It doesn't tell you which bumpswitch is pressed)

### 2.3 Whiteline Sensors

Firebird has 3 whiteline sensors. They are considered as left, middle and right whiteline sensors. Each whiteline sensor has a light transmitter and receiver. The emitted is reflected from the surface and received by the receiver. This analog value is converted in to a digital value. If this value is less than 10 then the surface is treated as white surface. Hence these sensors are modeled as valued input signals which can take integer value.

- **LEFT\_WHITELINE\_VALUE(integer)** Input signal getting the value of left whiteline sensor
- **MIDDLE\_WHITELINE\_VALUE(integer)** Input signal getting the value of middle whiteline sensor
- **RIGHT\_WHITELINE\_VALUE(integer)** Input signal getting the value of right whiteline sensor

## 2.4 Infrared Range Sensors

Firebird can accommodate 3 infrared range sensors(IR sensors). They are called as left, front and right IR sensors. These are also modeled as integer valued input signals. The signal returns the value from the obstacle in millimeters(mm).

- **LEFT\_IR\_VALUE(integer)** Input signal getting the value of left IR sensor
- **FRONT\_IR\_VALUE(integer)** Input signal getting the value of middle IR sensor
- **RIGHT\_IR\_VALUE(integer)** Input signal getting the value of right IR sensor

## 2.5 Battery Voltage

To know the current voltage of firebird's battery we can use **BATTERY\_VOLTAGE(integer)** signal. It is a integer valued input signal.

## 2.6 Stepper Motors

Firebird has 2 stepper motors on which it moves. we consider them as left and right motors. There are various output signals modeled to control these motors. They are

- **MOVE\_FWD** emitting this signal moves the firebird in forward direction
- **MOVE\_REV** emitting this signal moves the firebird in backward direction
- **MOVE\_LEFT** emitting this signal moves the firebird in left direction(Right motor velocity is zero here)
- **MOVE\_RIGHT** emitting this signal moves the firebird in right direction(Left motor velocity is zero here)
- **MOVE\_INPLACE\_LEFT** emitting this signal moves the firebird in left direction(Left motor rotates in backward direction and Right motor in reverse direction)
- **MOVE\_INPLACE\_RIGHT** emitting this signal moves the firebird in right direction(Right motor rotates in backward direction and Left motor in reverse direction)
- **MOTOR\_LEFT\_SPEED(integer)** An integer valued signal to set the speed of left motor
- **MOTOR\_RIGHT\_SPEED(integer)** An integer valued signal to set the speed of right motor speed

## 2.7 Shaft Encoders

These are used to measure the distance moved by Firebird. There are 2 shaft encoders. Each stepper motors has one shaft encoder. This is modeled as input signal.

- **MOTOR\_SHAFT\_L** Input signal for left shaft encoder
- **MOTOR\_SHAFT\_R** Input signal for right shaft encoder

To **move** the Firebird forward or back ward by some distance, we use these shaft encoder signals. For each input signal of shaft encoder, Firebird moves 2.9083 mm.

eg. To move Firebird forward by 100mm, we should wait for 35 shaft encoder signals.

Two types of **rotations** are possible for Firebird. They are

- Normal rotation
- Inplace rotation

and each rotation can be either in left or right direction.

In normal rotation mode, Firebird rotates 360 degrees for 236 signals of shaft encoder.

Whereas in inplace rotation mode it takes 115 signals of shaft encoder for 360 degrees.

eg. In normal left rotation mode only left motor moves, so to rotate fire bird left in normal rotation mode we should wait for left shaft encoder signals.

## 2.8 LCD

Firebird has a LCD which has 2 rows of display. Each row can display 16 characters. since the LCD can display both integers and characters, the signal type can be either integer or string.

- **LCD\_CLEAR** clears the LCD(erases the current data)
- **LCD\_DISPLAY\_1(string)** displays the given string on the first row of LCD(erases the current data on LCD and prints new data)
- **LCD\_DISPLAY\_2(string)** displays the given string on the second row of LCD
- **LCD\_DISPLAY\_INT\_1(integer)** displays the given integer on the first row of LCD(erases the current data on LCD and prints new data)
- **LCD\_DISPLAY\_INT\_2(integer)** displays the given integer on the second row of LCD

## 2.9 tick

*tick* is a special input signal in Esterel which is always present at every instant. For Firebird *tick* is modeled as a delay of 1 msec..

eg. *await 1000 tick*; results in a delay of 1 sec.

## 2.10 Sample Program

Here is a simple program. In this program, firebird waits for any of the bump switch to be pressed and as soon as it detects a bump press, it prints the total no.of bump presses till now on LCD.

```
module sample:
  input BUMPSWITCH_ANY;
  output LCD_DISPLAY_INT_1(integer);
  var count:integer in
    count := 0;
    every immediate BUMPSWITCH_ANY do
      count := count + 1;
      emit LCD_DISPLAY_INT_1(count);
    end every
  end var
end module
```

## 3 Kontroller Lab settings

### 3.1 Configure Project Settings

Go to “project” tab and choose “Configure Project”.

#### 3.1.1 Common tab settings

Figure 1 shows the common tab settings. use the following setting.

- CPU – ATmega128
- Clock – 11100000 Hz
- Select “Use built-in build system” radio button
- HEX file name – write your hex file name. eg. prog.hex

For every tab after setting, click on the “Set as default” button to make these settings default.

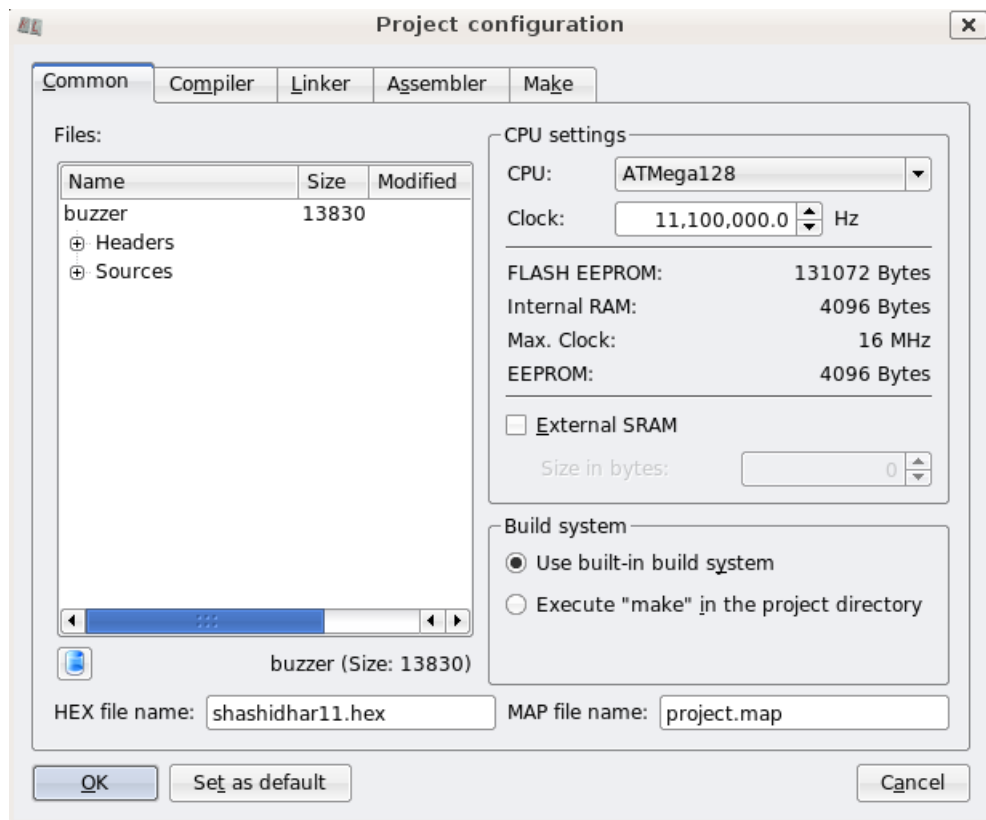


Figure 1: Project configuration – common tab settings

### 3.1.2 Compiler tab settings

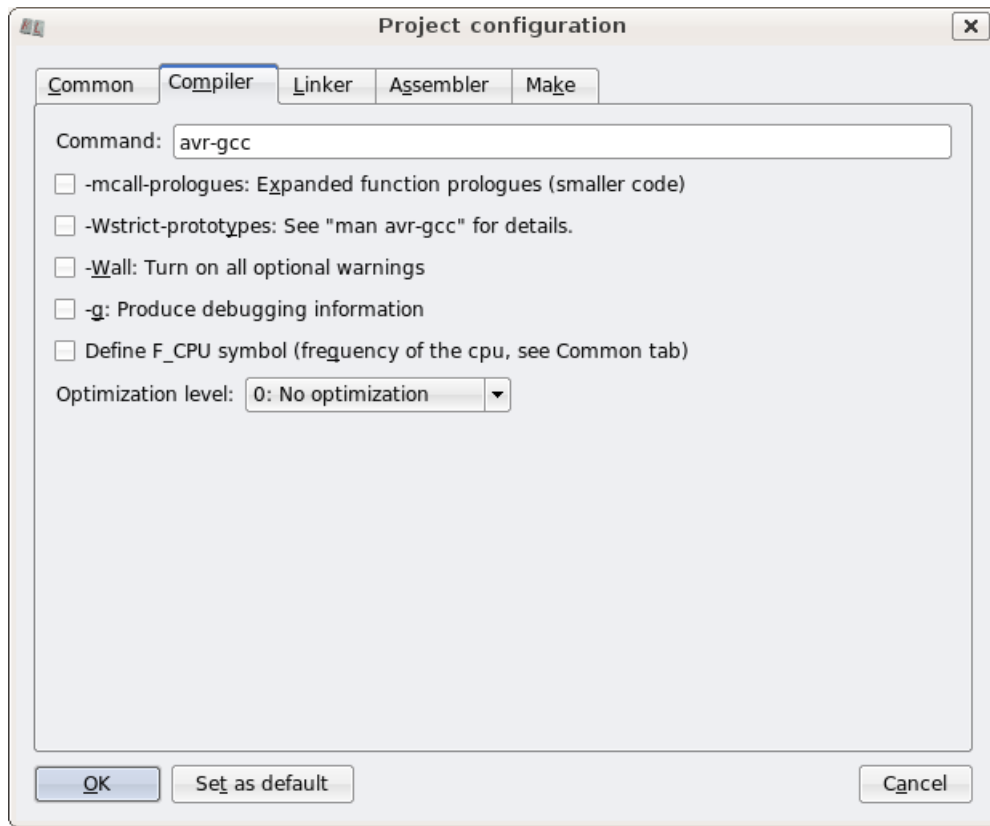


Figure 2: Project configuration – compiler tab settings

### 3.1.3 Linker tab settings

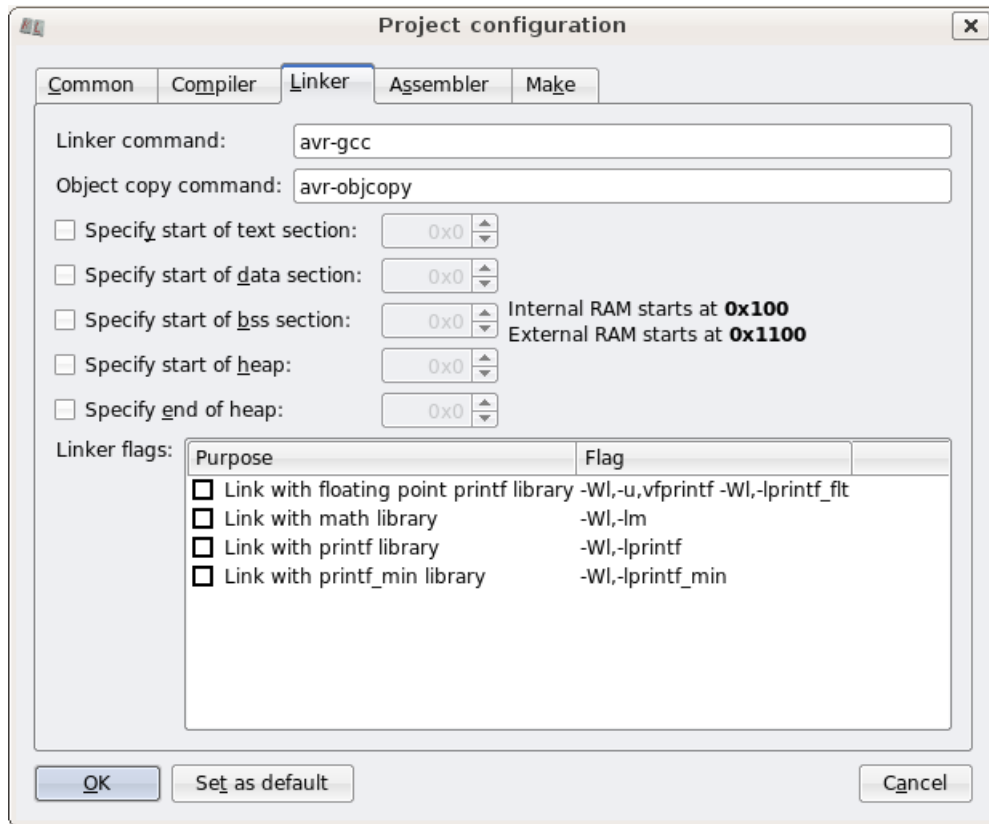


Figure 3: Project configuration – Linker tab settings

### 3.1.4 Assembler tab settings

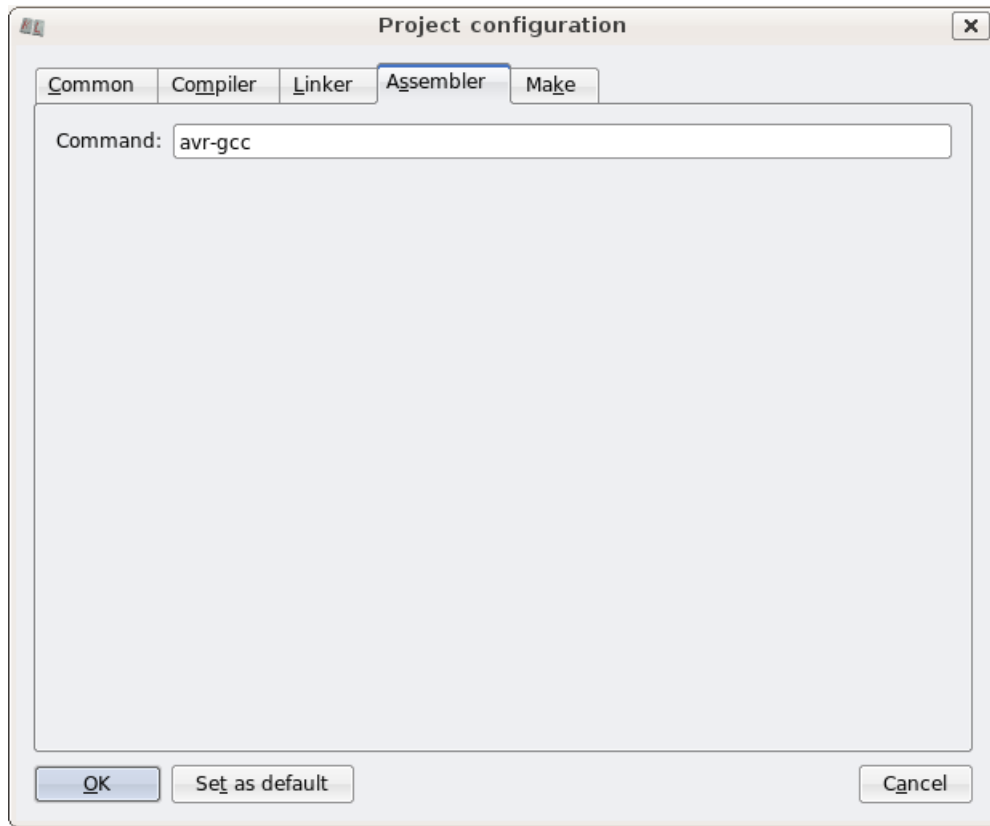


Figure 4: Project configuration – Assembler tab settings

### 3.1.5 Make tab settings

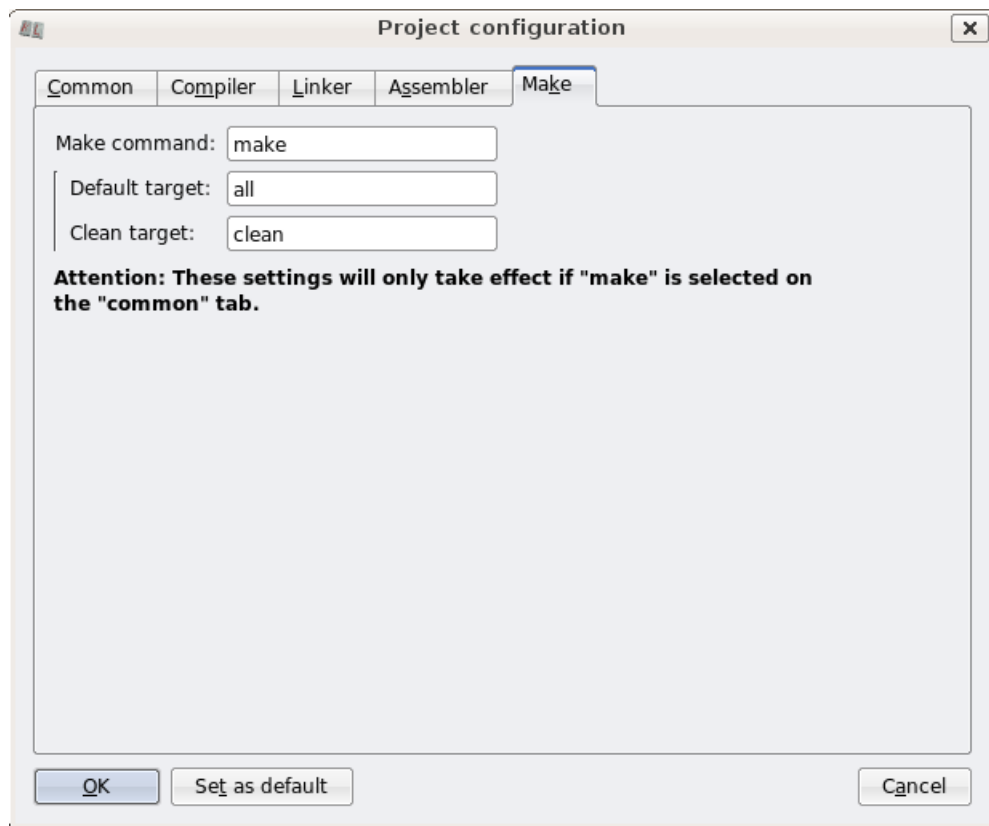


Figure 5: Project configuration – Make tab settings

## 3.2 Configure Programmer Settings

Go to “project” tab and choose “Configure Programmer”.

### 3.2.1 Choose Programmer tab settings

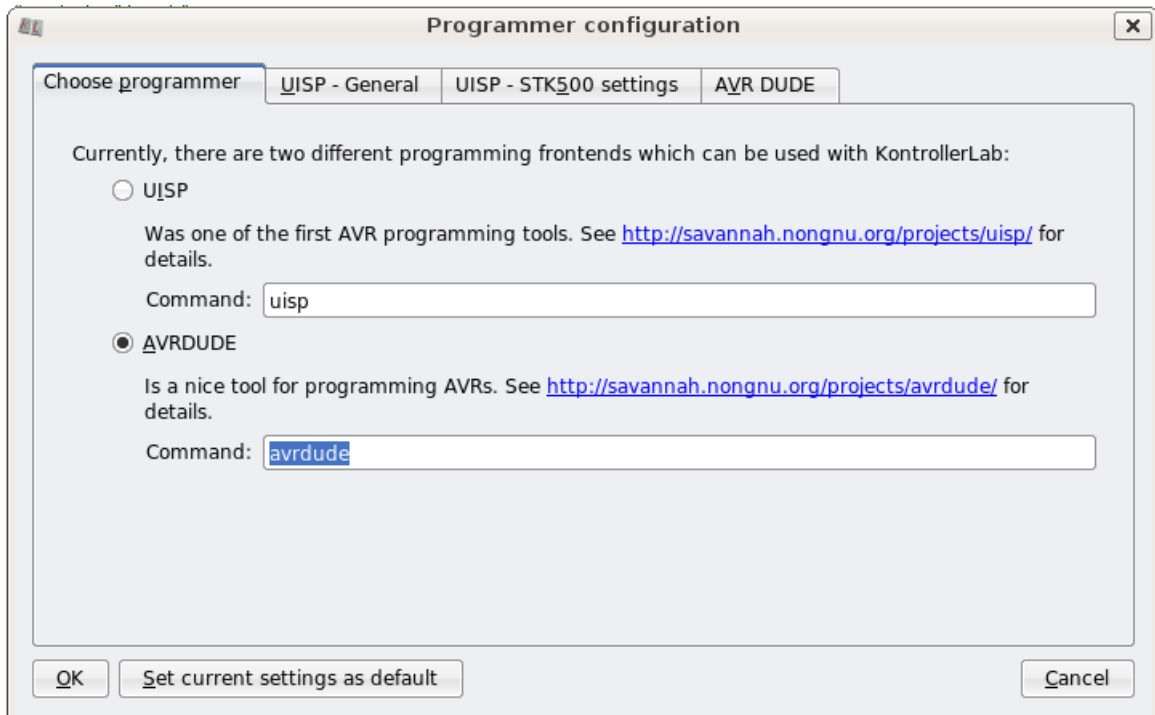


Figure 6: Programmer Configuration – choose programmer tab settings

### 3.2.2 UISP-General tab settings

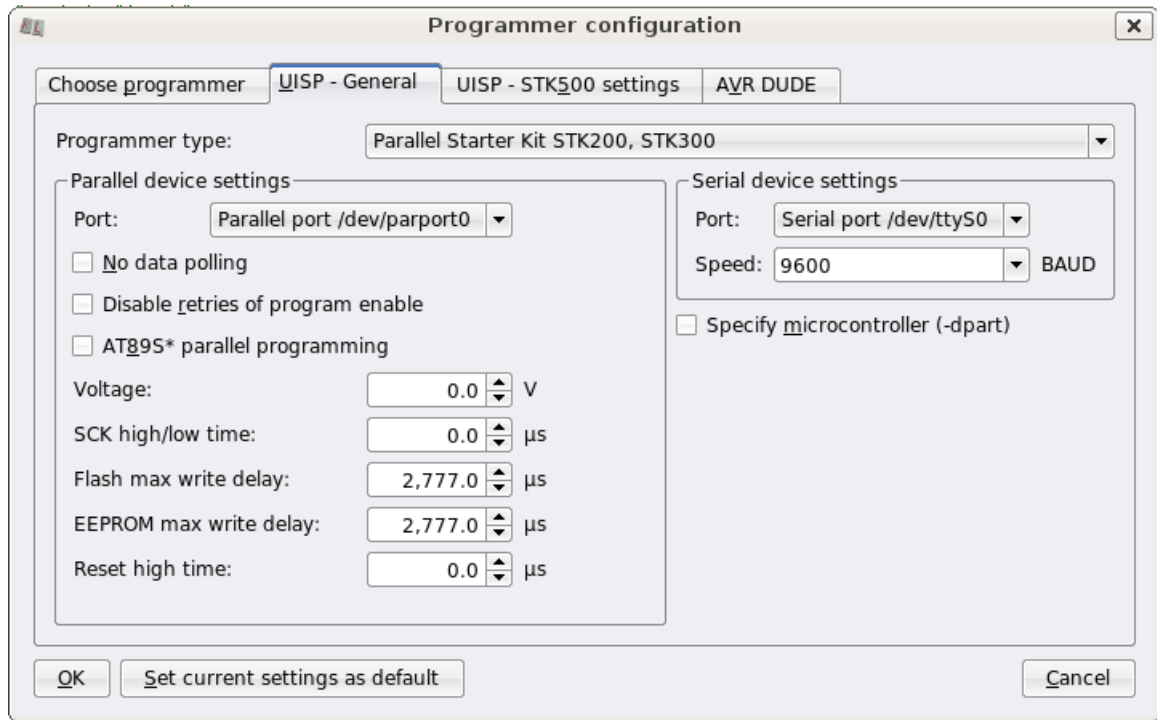


Figure 7: Programmer Configuration – UISP-General tab settings

### 3.2.3 UISP-STK500 tab settings

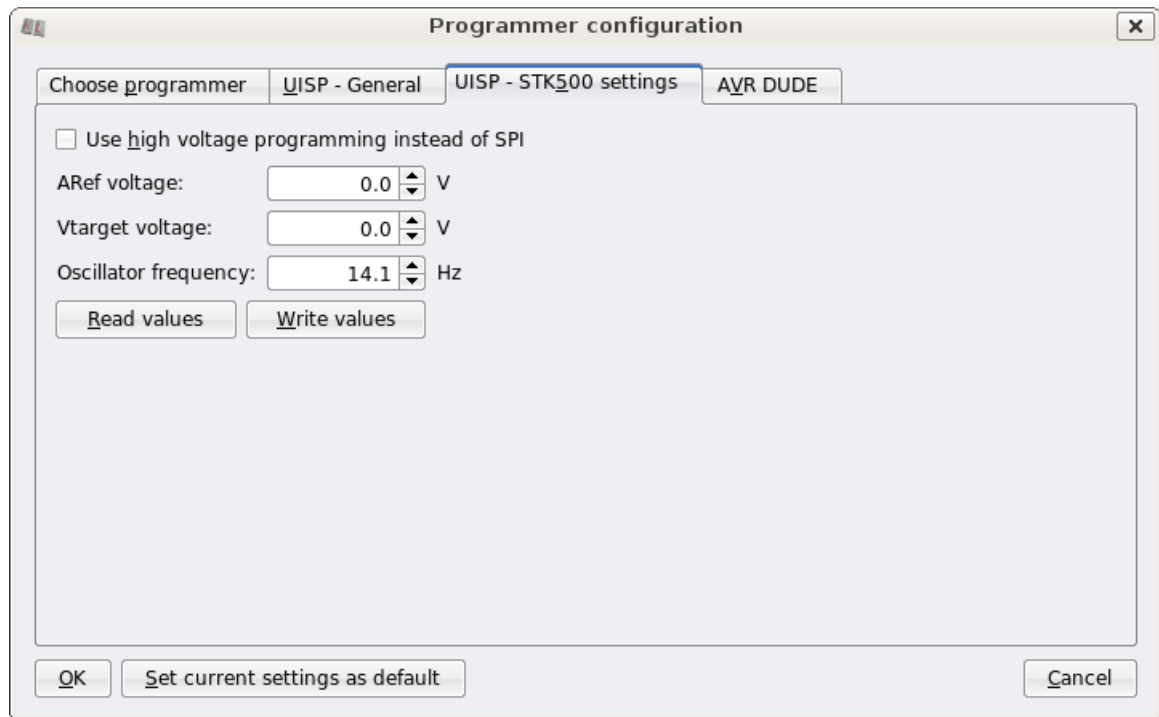


Figure 8: Programmer Configuration – UISP-STK500 tab settings

### 3.2.4 AVR DUDE tab settings

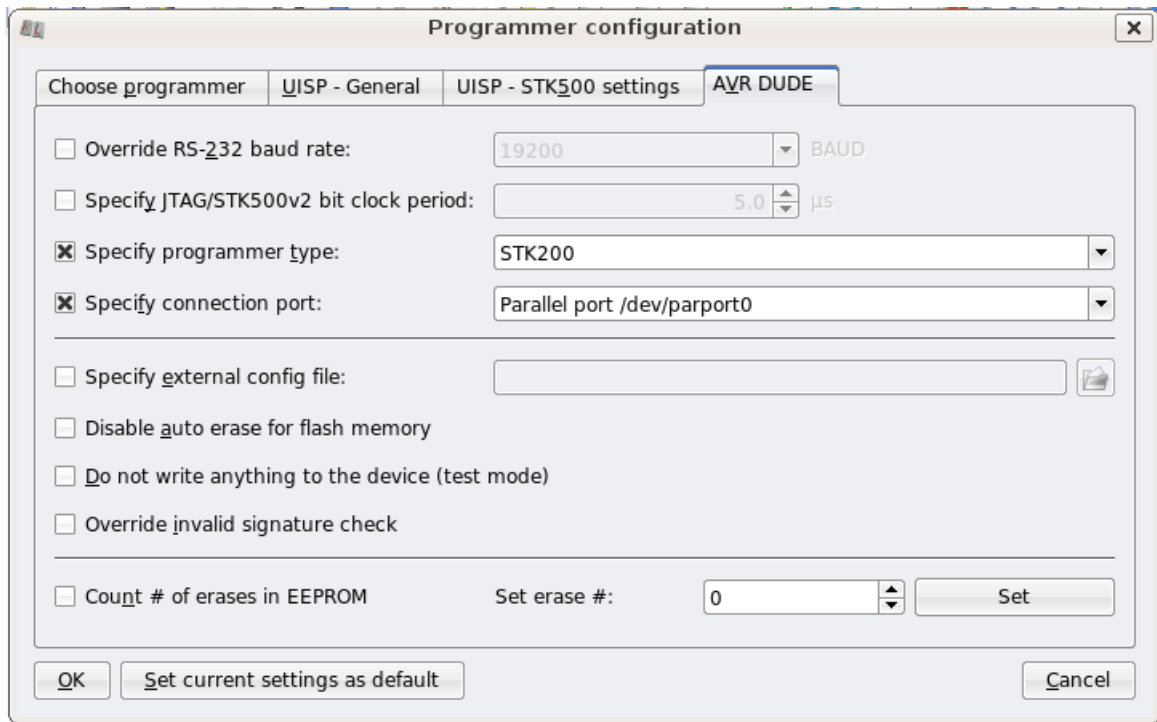


Figure 9: Programmer Configuration – AVR DUDE tab settings