This directory contains the Optiboot small bootloader for AVR

microcontrollers, somewhat modified specifically for the Arduino

environment.

Optiboot is more fully described here: http://code.google.com/p/optiboot/

and is the work of Peter Knight (aka Cathedrow), building on work of Jason P

Kyle, Spiff, and Ladyada. Arduino-specific modification are by Bill

Westfield (aka WestfW)

Arduino-specific issues are tracked as part of the Arduino project

at http://code.google.com/p/arduino

------------------------------------------------------------

Building optiboot for Arduino.

Production builds of optiboot for Arduino are done on a Mac in "unix mode"

using CrossPack-AVR-20100115. CrossPack tracks WINAVR (for windows), which

is just a package of avr-gcc and related utilities, so similar builds should

work on Windows or Linux systems.

One of the Arduino-specific changes is modifications to the makefile to

allow building optiboot using only the tools installed as part of the

Arduino environment, or the Arduino source development tree. All three

build procedures should yield identical binaries (.hex files) (although

this may change if compiler versions drift apart between CrossPack and

the Arduino IDE.)

Building Optiboot in the Arduino IDE Install.

Work in the .../hardware/arduino/bootloaders/optiboot/ and use the

"omake <targets>" command, which just generates a command that uses

the arduino-included "make" utility with a command like:

make OS=windows ENV=arduino <targets>

or make OS=macosx ENV=arduino <targets>

On windows, this assumes you're using the windows command shell. If

you're using a cygwin or mingw shell, or have one of those in your

path, the build will probably break due to slash vs backslash issues.

On a Mac, if you have the developer tools installed, you can use the

Apple-supplied version of make.

The makefile uses relative paths ("../../../tools/" and such) to find

the programs it needs, so you need to work in the existing optiboot

directory (or something created at the same "level") for it to work.

Building Optiboot in the Arduino Source Development Install.

In this case, there is no special shell script, and you're assumed to

have "make" installed somewhere in your path.

Build the Arduino source ("ant build") to unpack the tools into the

expected directory.

Work in Arduino/hardware/arduino/bootloaders/optiboot and use

make OS=windows ENV=arduinodev <targets>

or make OS=macosx ENV=arduinodev <targets>

Programming Chips Using the \_isp Targets

The CPU targets have corresponding ISP targets that will actuall

program the bootloader into a chip. "atmega328\_isp" for the atmega328,

for example. These will set the fuses and lock bits as appropriate as

well as uploading the bootloader code.

ISP Targets in Version 5.0 and later:

The isp targets are now built using a separate "Makefile.isp" makefile,

which should make modification easier and more obvious. This also fixes

the atmega8\_isp target problem mentioned below. The default

configuration assumes an ArduinoISP setup, but you will probably need to

update at least the serial port, since those are different for each

Arduino board and/or system/

ISP Targets in Version 4.6 and earlier:

The older makefiles default to using a USB programmer, but you can use a

serial programmer like ArduinoISP by changing the appropriate variables

when you invoke make:

make ISPTOOL=stk500v1 ISPPORT=/dev/tty.usbserial-A20e1eAN \

ISPSPEED=-b19200 atmega328\_isp

The "atmega8\_isp" target does not currently work, because the mega8

doesn't have the "extended" fuse that the generic ISP target wants to

pass on to avrdude. You'll need to run avrdude manually.

Standard Targets

I've reduced the pre-built and source-version-controlled targets

(.hex and .lst files included in the git repository) to just the

three basic 16MHz targets: atmega8, atmega16, atmega328.