Atmega bootloader programmer

This subject is now closed. Start a new subject Refresh page

Posted by **Nick Gammon** Australia (18,356 posts) bio *Forum Administrator*

Date Sun 06 May 2012 09:52 PM (UTC) [quote]

Amended on Fri 17 Aug 2012 12:44 AM (UTC) by Nick Gammon

Message This sketch was inspired by the Optiloader sketch written for the Arduino. However it is a total rewrite, in order to accomodate the Mega2560 board, which the original one did not handle, due to the larger address space.

Code

http://gammon.com.au/Arduino/Atmega_Board_Programmer.zip

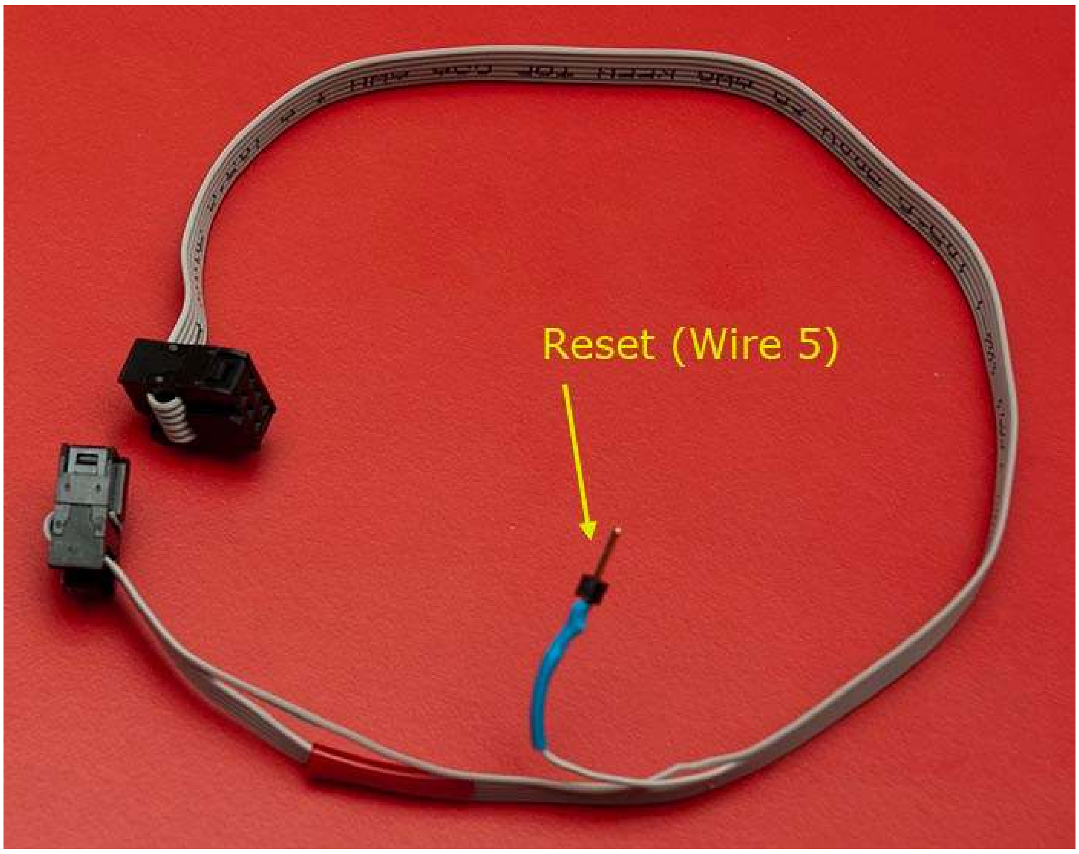
Download the above file, unzip it, and place the resulting folder into your Arduino "sketches" folder.

If you try to compile under older versions of the Arduino IDE, which do not support the F macro, you will get errors. You can get a copy of that here:

```
http://arduiniana.org/libraries/flash/
```

Programming cable

For boards which have an ICSP header, it helps to make up a programming cable (although not essential). You can do this by getting a 6-pin IDC cable, and cutting the 5th wire out of circuit and soldering a pin onto it, like this:



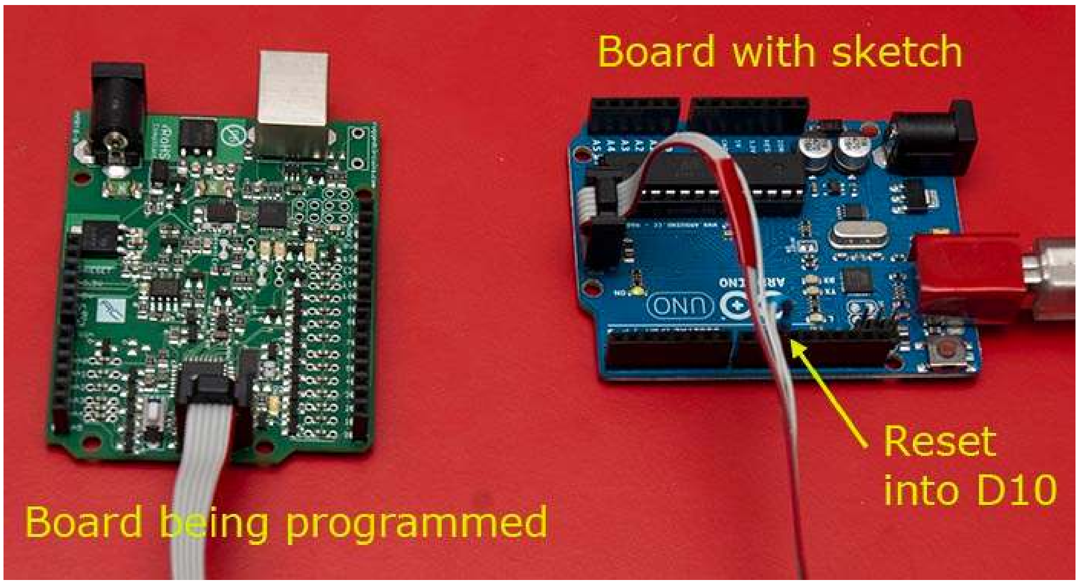
This wire (the reset signal) gets plugged into D10 on the Arduino with the programming sketch on it, as in the images below.

If you don't have such a cable, just connect together the pins as described below.

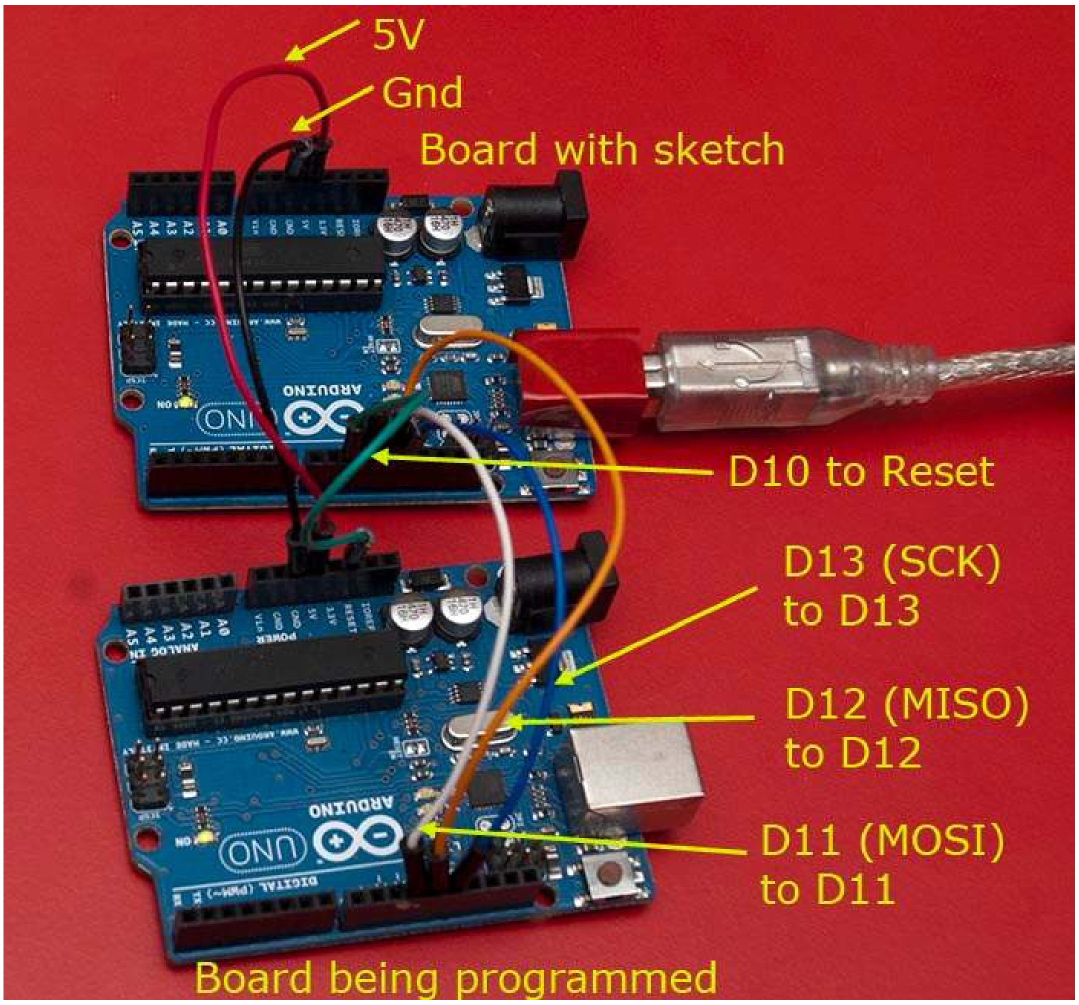
Programming a board

Upload above sketch to the "programming" Uno. Connect programming cable as shown.

Wiring for Uno and similar



(Ruggeduino depicted with programming cable)



(Uno depicted with "manual" programming cable)

```
**Arduino Uno      Target Uno**

D10  (SS)            Reset
D11  (MOSI)          D11
D12  (MISO)          D12
D13  (SCK)           D13

Gnd                  Gnd
+5V                  +5V
```

Example output for Uno

```
Atmega chip programmer.
Written by Nick Gammon.
Entered programming mode OK.
Signature = 0x1E 0x95 0x0F
Processor = ATmega328P
Flash memory size = 32768 bytes.
LFuse = 0xFF
HFuse = 0xDE
EFuse = 0xFD
Lock byte = 0xCF
Bootloader address = 0x7E00
Bootloader length = 512 bytes.
Type 'G' to program the chip with the bootloader ...
```

Once you see the above, type "G" into the serial monitor to commence programming ...

```
Erasing chip ...
Writing bootloader ...
Committing page starting at 0x7E00
Committing page starting at 0x7E80
Committing page starting at 0x7F00
Committing page starting at 0x7F80
Written.
Verifying ...
No errors found.
Writing fuses ...
LFuse = 0xFF
HFuse = 0xDE
EFuse = 0xFD
Lock byte = 0xCF
Done.
Type 'C' when ready to continue with another chip ...
```

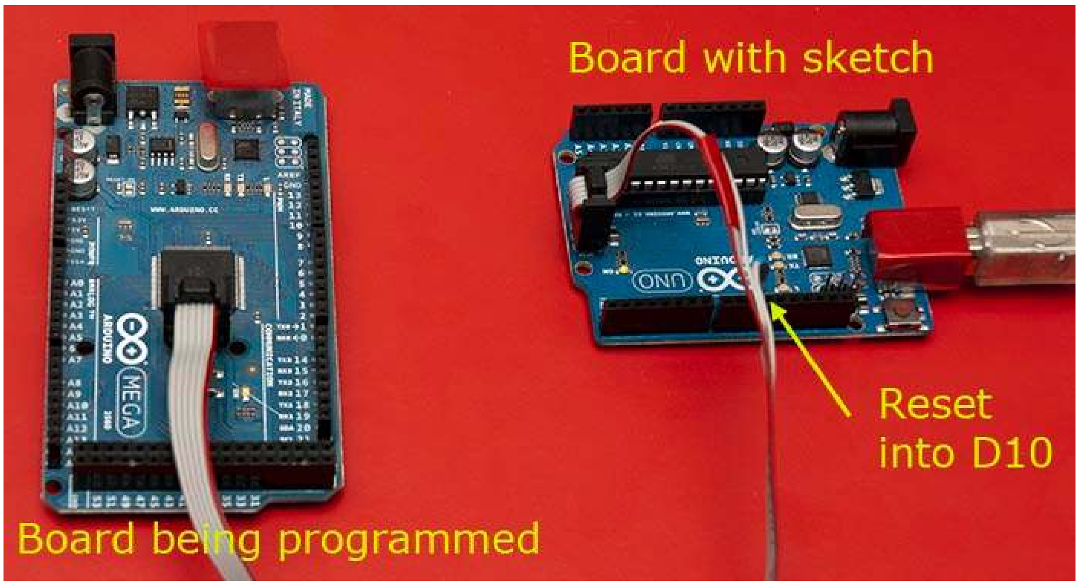
This takes one second.

[EDIT] The more recent versions of the sketch ask this question:

```
Type 'L' to use Lilypad (8 MHz) loader, or 'U' for Uno (16 MHz) loader ...
```

Type "U" if you are using a Uno (or any board running at 16 MHz) or "L" if you are using an 8 MHz board.

Wiring for Mega2560 and similar



```
Arduino Uno       Target Mega2560

D10  (SS)            Reset
D11  (MOSI)          D51
D12  (MISO)          D50
D13  (SCK)           D52

Gnd                  Gnd
+5V                  +5V
```

Example output for Mega2560

```
Atmega chip programmer.
Written by Nick Gammon.
Entered programming mode OK.
Signature = 0x1E 0x98 0x01
Processor = ATmega2560
```

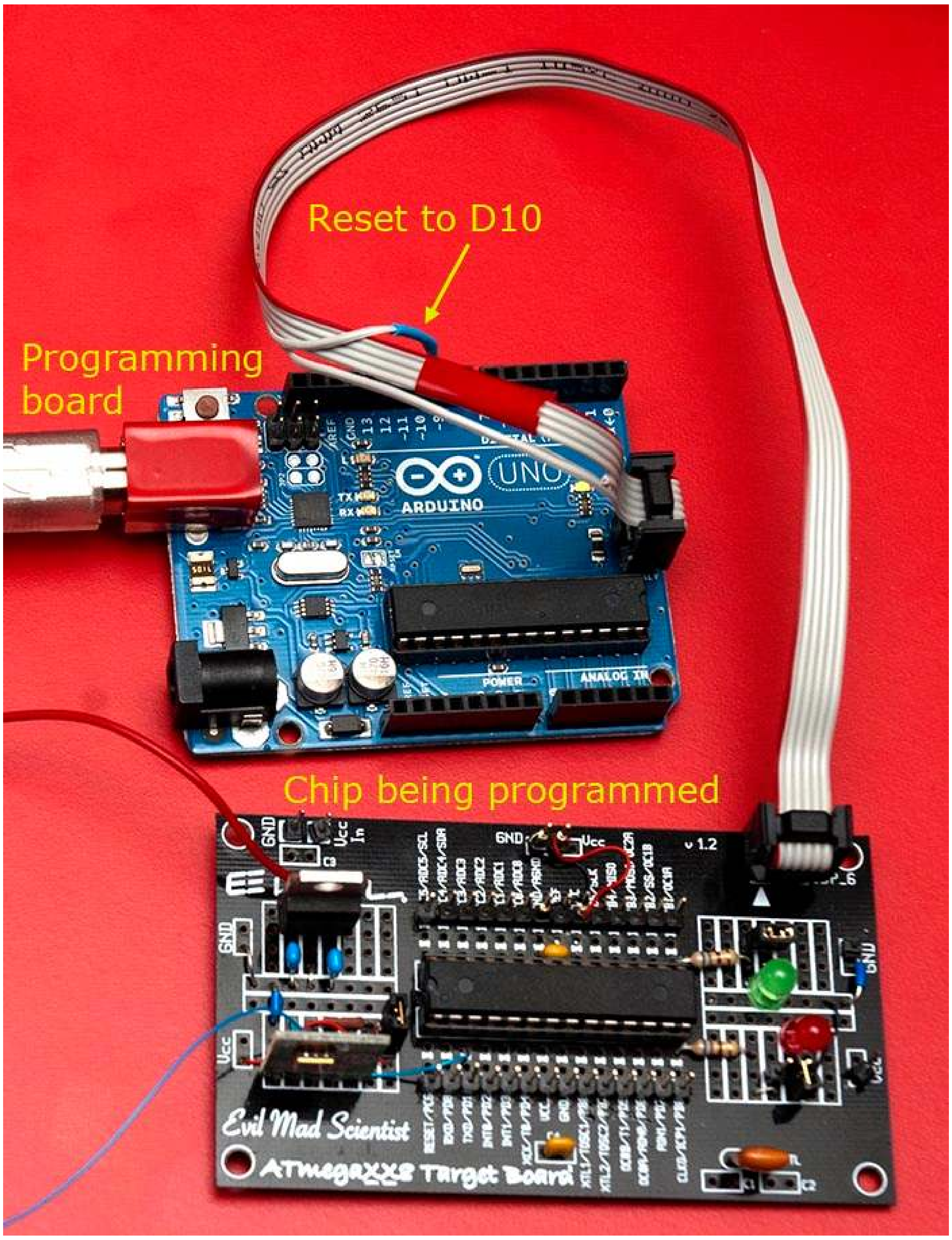
```
Flash memory size = 262144 bytes.
LFuse = 0xFF
HFuse = 0xD8
EFuse = 0xFD
Lock byte = 0xCF
Bootloader address = 0x3E000
Bootloader length = 8192 bytes.
Type 'G' to program the chip with the bootloader ...
```

Once you see the above, type "G" into the serial monitor to commence programming ...

```


Erasing chip ...
Writing bootloader ...
Committing page starting at 0x3E000
Committing page starting at 0x3E100
Committing page starting at 0x3E200
Committing page starting at 0x3E300
Committing page starting at 0x3E400
Committing page starting at 0x3E500
Committing page starting at 0x3E600
Committing page starting at 0x3E700
Committing page starting at 0x3E800
Committing page starting at 0x3E900
Committing page starting at 0x3EA00
Committing page starting at 0x3EB00
Committing page starting at 0x3EC00
Committing page starting at 0x3ED00
Committing page starting at 0x3EE00
Committing page starting at 0x3EF00
Committing page starting at 0x3F000
Committing page starting at 0x3F100
Committing page starting at 0x3F200
Committing page starting at 0x3F300
Committing page starting at 0x3F400
Committing page starting at 0x3F500
Committing page starting at 0x3F600
Committing page starting at 0x3F700
Committing page starting at 0x3F800
Committing page starting at 0x3F900
Committing page starting at 0x3FA00
Committing page starting at 0x3FB00
Committing page starting at 0x3FC00
Committing page starting at 0x3FD00
Committing page starting at 0x3FE00
Committing page starting at 0x3FF00
Written.
Verifying ...
No errors found.
Writing fuses ...
LFuse = 0xFF
HFuse = 0xD8
EFuse = 0xFD
Lock byte = 0xCF
Done.
Type 'C' when ready to continue with another chip ...
```

This takes three seconds.

Programming a "bare bones" board

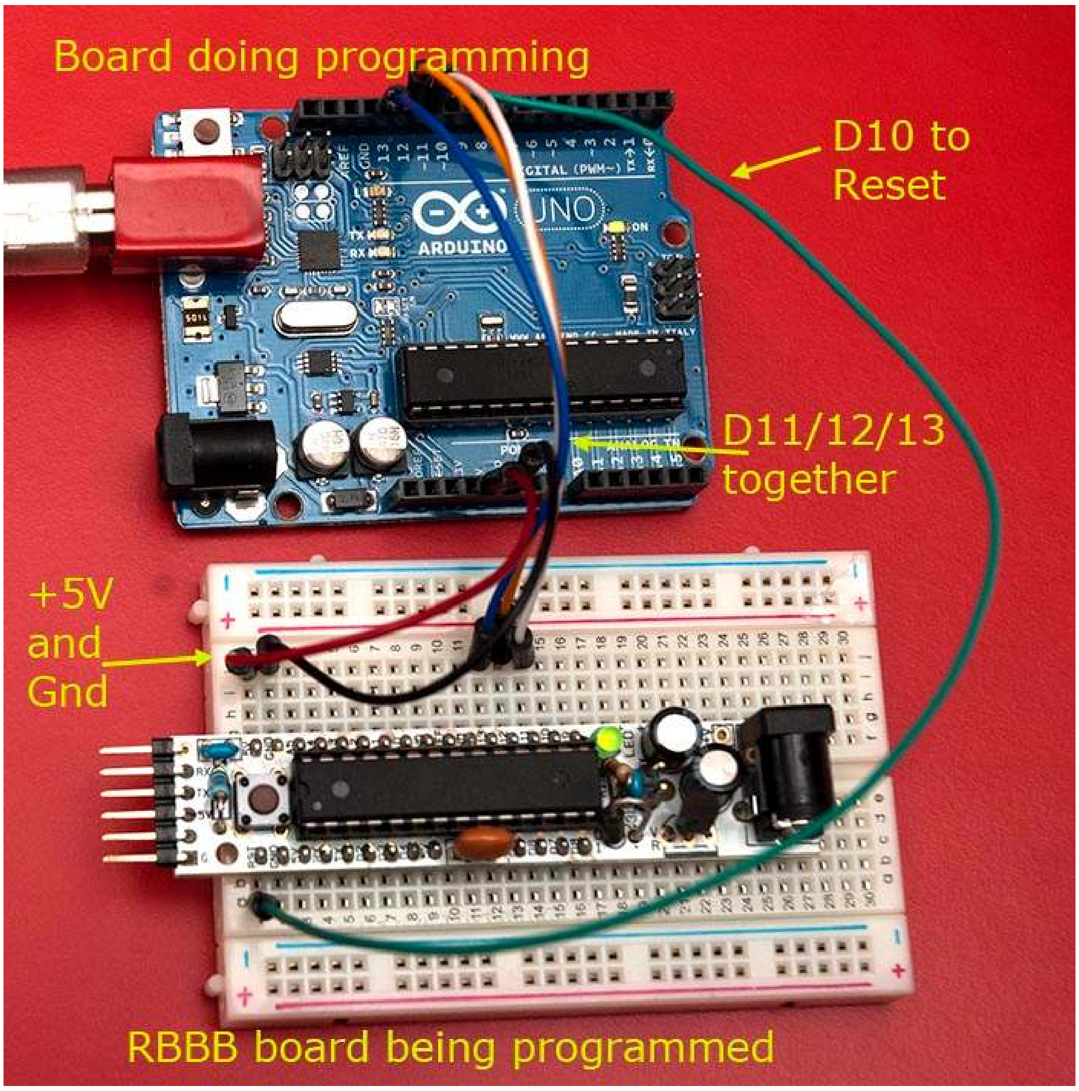


```
Atmega chip programmer.
Written by Nick Gammon.
Entered programming mode OK.
Signature = 0x1E 0x95 0x0F
Processor = ATmega328P
Flash memory size = 32768 bytes.
LFuse = 0xFF
HFuse = 0xDA
EFuse = 0xFC
Lock byte = 0xFF
Bootloader address = 0x7E00
Bootloader length = 512 bytes.
Type 'G' to program the chip with the bootloader ...
Erasing chip ...
Writing bootloader ...
Committing page starting at 0x7E00
Committing page starting at 0x7E80
Committing page starting at 0x7F00
Committing page starting at 0x7F80
Written.
Verifying ...
No errors found.
Writing fuses ...
LFuse = 0xFF
HFuse = 0xDE
EFuse = 0xFD
```

```
Lock byte = 0xCF
Done.
Type 'C' when ready to continue with another chip ...
```

Note that the fuses were changed after the board was programmed.

Programming a RBB (really bare bones board)



```
Atmega chip programmer.
Written by Nick Gammon.
Entered programming mode OK.
Signature = 0x1E 0x95 0x0F
Processor = ATmega328P
Flash memory size = 32768 bytes.
LFuse = 0xFF
HFuse = 0xDA
EFuse = 0xFD
Lock byte = 0xCF
Bootloader address = 0x7E00
Bootloader length = 512 bytes.
Type 'G' to program the chip with the bootloader ...
Erasing chip ...
Writing bootloader ...
Committing page starting at 0x7E00
Committing page starting at 0x7E80
Committing page starting at 0x7F00
Committing page starting at 0x7F80
Written.
Verifying ...
No errors found.
Writing fuses ...
LFuse = 0xFF
HFuse = 0xDE
EFuse = 0xFD
Lock byte = 0xCF
```

```
Done.
Type 'C' when ready to continue with another chip ...
```

Adding other bootloaders

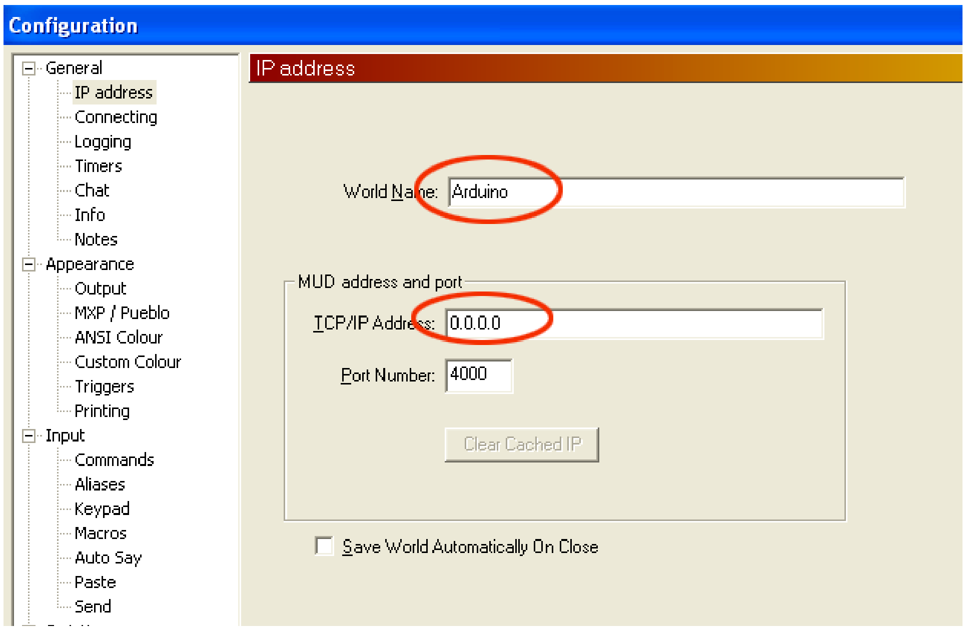
In order to convert the .hex bootloader files on disk for use with this sketch I ran this Lua script:

```
-- Do MD5 sumcheck of Arduino bootloader .hex file
-- Nick Gammon
-- 5 May 2012

require "getlines"

loader = nil
adder = 0

-- given a start address, deduce where the bootloader ends
end_addresses = {
  [0x1C00] = 0x2000,
  [0x1D00] = 0x2000,
  [0x1E00] = 0x2000,
  [0x3800] = 0x4000,
  [0x3E00] = 0x4000,
  [0x7000] = 0x8000,
  [0x7800] = 0x8000,
  [0x7E00] = 0x8000,
  [0xF800] = 0x10000,
  [0x1F000] = 0x20000,
  [0x1FC00] = 0x20000,
  [0x3E000] = 0x40000,
  }

function process (size, address, rectype, data)

  size = tonumber (size, 16)
  address = tonumber (address, 16) + adder
  rectype = tonumber (rectype)
  local binarydata = utils.fromhex (data)

  if rectype == 2 then  -- Extended Segment Address Record
    adder = tonumber (data, 16) * 16  -- high order address byte
  elseif rectype == 0 then -- data record
    if loader == nil then
      start_address = address
      end_address = end_addresses [address]
      if end_address == nil then
        ColourNote ("red", "", "Don't know end address for " .. bit.tostring (address, 16))
        ColourNote ("red", "", "Please add to table: end_addresses")
        error "Cannot continue"
      end -- if end address not found

      -- work out loader length
      length = end_address - address
      -- pre-fill with 0xFF in case not every byte supplied
      loader = string.rep ("\255", length)
      print (string.format ("// Loader start: %X, length: %i", address, length))
    end -- no loader yet

    -- insert data over where the 0xFF was
    if address >= start_address and (address + size) <= end_address then
      loader = loader:sub (1, address - start_address) ..
               binarydata ..
               loader:sub (address - start_address + size + 1, length)
    else
      ColourNote ("red", "", "Address " .. bit.tostring (address, 16) .. " out of expected range.")
    end  -- if in range
  end -- if

end -- function process

print (string.rep ("-", 60))

-- get bootloader file
filename = utils.filepicker ("Choose a bootloader", nil, "hex", { hex = "Hex files" })

-- none chosen, give up
if not filename then return end

-- show file
local fn = string.match (filename, "\\([^\\]+)$")
print ("// File = ", fn)

-- process each line
for line in io.lines (filename) do
  size, address, rectype, data = string.match (line, "^:(%x%x)(%x%x%x%x)(%x%x)(%x+)%s*$")
  if size then
    process (size, address, rectype, data:sub (1, -3))
  else
```

```
      ColourNote ("red", "", "Discarded line: " .. line)
    end -- if
  end -- for loop

  --print (utils.tohex (loader))

  -- sumcheck it
  Tell ("// MD5 sum = ")
  md5sum = utils.tohex (utils.md5 (loader))
  print ((string.gsub (md5sum, "(%x%x)", "%1 ")))

  print ""

  --      show bootloader in hex

  -- convert into C array
  print (string.format ("byte PROGMEM %s [] = {",
         string.gsub (fn, "[^%a%d]", "_")))
  for i = 1, #loader do
    Tell (string.format ("0x%02X, ", loader:sub (i, i):byte ()))
    if (i - 1) % 16 == 15 then print "" end
  end -- of for each byte
  print (string.format ("}; // end of %s", string.gsub (fn, "[^%a%d]", "_")))
```

Because it uses a few special features (like a file picker, MD5 sum calculator etc.) you need to run this from within MUSHclient ... a MUD game client available for free download from this site.

Just download MUSHclient (see the downloads page) and install. Then make a "dummy" world file (File menu -> New World) and type in "Arduino" as the world name and "0.0.0.0" as the TCP/IP address, as shown:



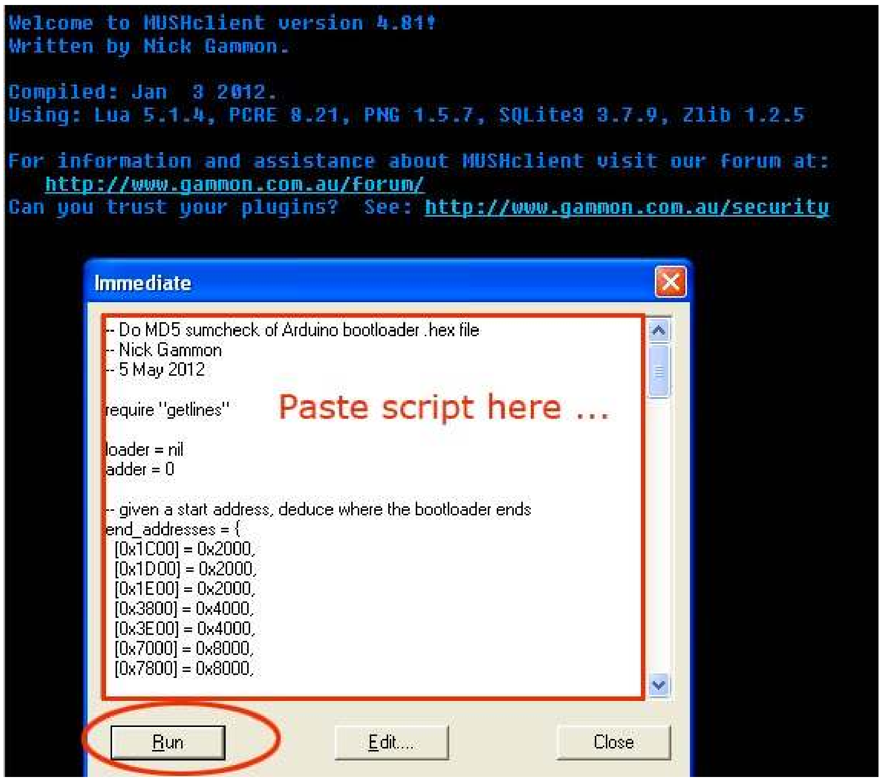
World Name: Arduino

TCP/IP Address: 0.0.0.0

Port Number: 4000

☐ Save World Automatically On Close

Close the world configuration, and then type Ctrl+I to open the Immediate programming window (Game menu -> Immediate) and paste the above script into it, then hit Run, as shown:



```
Welcome to MUSHclient version 4.81!
Written by Nick Gammon.

Compiled: Jan  3 2012.
Using: Lua 5.1.4, PCRE 8.21, PNG 1.5.7, SQLite3 3.7.9, Zlib 1.2.5

For information and assistance about MUSHclient visit our forum at:
   http://www.gammon.com.au/forum/
Can you trust your plugins?  See: http://www.gammon.com.au/security
```

```
-- Do MD5 sumcheck of Arduino bootloader .hex file
-- Nick Gammon
-- 5 May 2012

require "getlines"

loader = nil
adder = 0

-- given a start address, deduce where the bootloader ends
end_addresses = {
  [0x1C00] = 0x2000,
  [0x1D00] = 0x2000,
  [0x1E00] = 0x2000,
  [0x3800] = 0x4000,
  [0x3E00] = 0x4000,
  [0x7000] = 0x8000,
  [0x7800] = 0x8000,
```

A file-picker will open, you can navigate to the bootloader .hex file that you want to process, and then (all being well) the appropriate hex codes will be shown in the main window. Close the Immediate window, and then just copy and paste into your bootloader sketch (make a separate xxxx.h tab like I did for the other bootloaders).

You will then need to fill in the appropriate figures into the signatures table, such as the bootloader start address, size, programming page size (see the datasheet) and the correct fuse and lock bits. The fuse bytes will probably be the ones mentioned in the boards.txt file in the Arduino environment, for the chip in question.

Checking what bootloader is installed

The thread below describes a sketch that does a MD5 sumcheck of your bootloader, so you can see what bootloader you have installed:

http://www.gammon.com.au/forum/?id=11633

- Nick Gammon

www.gammon.com.au, www.mushclient.com

top

The dates and times for posts above are shown in Universal Co-ordinated Time (UTC).

To show them in your local time you can join the forum, and then set the 'time correction' field in your profile to the number of hours difference between your location and UTC time.

2,416 views.

This subject is now closed. Start a new subject Refresh page

Go to topic: (Choose topic) Go Search the forum

top