This repository Search Explore Gist Blog Help beyondszine

nickgammon / **arduino_sketches**

Watch 22 Star 38 Fork 14

Publicly-released sketches for the Arduino microprocessor

61 commits **1** branch **6** releases **1** contributor

branch: **master** **arduino_sketches** / +

Turned programming mode off after uploading bootloader

nickgammon authored 17 days ago latest commit 88c7ed2e5c

Atmega_Board_Detector	Added MD5 sum for Uno Atmega16U2 (USB) bootloader	19 days ago
Atmega_Board_Programmer	Turned programming mode off after uploading bootloader	17 days ago
Atmega_Fuse_Calculator	Fixed compiling problems under IDE 1.5.8	2 months ago
Atmega_Hex_Uploader	Added comment about Atmega32U4 flash page size	25 days ago
Atmega_Self_Read_Signature	Initial commit	a year ago
.gitignore	Initial load	3 years ago
README.md	Added more information to readme file	18 days ago

Code
Issues 1
Pull Requests 0
Wiki
Pulse
Graphs

HTTPS clone URL
https://github.com

You can clone with HTTPS, SSH, or Subversion.

Download ZIP

README.md

arduino_sketches

Publicly-released sketches for the Arduino microprocessor.

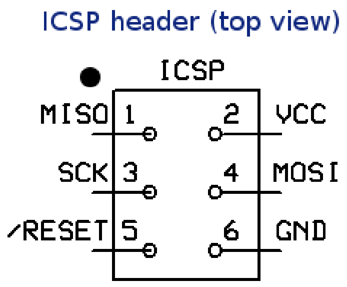
Wiring

All sketches except for Atmega_Hex_Uploader and Atmega_Self_Read_Signature:

Connect hardware SPI pins together as below:

- MISO to MISO
- MOSI to MOSI
- SCK to SCK
- Vcc to Vcc
- Gnd to Gnd
- Reset on target board to D10 on programming board

These pins are generally exposed on the ICSP header of most boards (except "breadboard" boards).



For "breadboard" boards you will need to read the datasheet for the appropriate chip to find which pins to use.

Atmega_Board_Detector

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

This uses one Arduino to detect the signature, fuses, and bootloader of another one.

Only some Arduinos are supported to run the sketch. It has been tested on a Uno, Mega2560 and Leonardo.

It sumchecks the bootloader so you can quickly see if a particular one is installed. Some bootloader sumchecks are known and the bootloader "name" reported if found.

Example of use:

```
Atmega chip detector.
Written by Nick Gammon.
Version 1.11
Compiled on Nov  4 2014 at 11:10:33
Entered programming mode OK.
Signature = 1E 95 0F
Processor = ATmega328P
Flash memory size = 32768 bytes.
LFuse = FF
HFuse = DE
EFuse = FD
Lock byte = CF
Clock calibration = 8D
Bootloader in use: Yes
EEPROM preserved through erase: No
Watchdog timer always on: No
Bootloader is 512 bytes starting at 7E00

Bootloader:

7E00: 11 24 84 B7 14 BE 81 FF F0 D0 85 E0 80 93 81 00
7E10: 82 E0 80 93 C0 00 88 E1 80 93 C1 00 86 E0 80 93
...
7FE0: 11 50 E9 F7 F2 DF 1F 91 08 95 80 E0 E8 DF EE 27
7FF0: FF 27 09 94 FF FF FF FF FF FF FF FF FF FF 04 04

MD5 sum of bootloader = FB F4 9B 7B 59 73 7F 65 E8 D0 F8 A5 08 12 E7 9F
Bootloader name: optiboot_atmega328

First 256 bytes of program memory:

0: 0C 94 52 05 0C 94 7A 05 0C 94 7A 05 0C 94 7A 05
10: 0C 94 7A 05 0C 94 7A 05 0C 94 7A 05 0C 94 7A 05
...
```

Atmega_Fuse_Calculator

See forum post: http://www.gammon.com.au/forum/?id=11653

Only some Arduinos are supported to run the sketch. It has been tested on a Uno, Mega2560 and Leonardo.

Similar to the Atmega_Board_Detector sketch, this reads a target board's fuses, and displays which fuses are set in a nicer interface, for example:

```
  External Reset Disable.................. [ ]
  Debug Wire Enable....................... [ ]
  Enable Serial (ICSP) Programming........ [X]
  Watchdog Timer Always On................ [ ]
  Preserve EEPROM through chip erase...... [ ]
  Boot into bootloader.................... [X]
  Divide clock by 8....................... [ ]
  Clock output............................ [ ]
```

Atmega_Self_Read_Signature

See forum post: http://www.gammon.com.au/forum/?id=11633&reply=2#reply2

Similar to the Atmega_Board_Detector sketch this "self-detects" a signature, so an Arduino can report back its own fuses, and bootloader MD5 sum.

Example of use:

```
  Signature detector.
  Written by Nick Gammon.
  Signature = 1E  95  0F
  Fuses
  Low = FF High = D6 Ext = FD Lock = CF

  Processor = ATmega328P
  Flash memory size = 32768
  Bootloader in use: Yes
  EEPROM preserved through erase: Yes
  Watchdog timer always on: No
  Bootloader is 512 bytes starting at 7E00
```

Note: Depending on the fuse settings, it may not be able to read the bootloader.

The Atmega_Self_Read_Signature sketch does not require any additional wiring.

Atmega_Board_Programmer

See forum post: http://www.gammon.com.au/forum/?id=11635

This will re-flash the bootloader in selected chips.

Only some Arduinos are supported to run the sketch. It has been tested on a Uno, Mega2560 and Leonardo.

Supported target chips are:

- Atmega8 (1024 bytes)
- Atmega168 Optiboot (512 bytes)
- Atmega328 Optiboot (for Uno etc. at 16 MHz) (512 bytes)
- Atmega328 (8 MHz) for Lilypad etc. (2048 bytes)
- Atmega32U4 for Leonardo (4096 bytes)
- Atmega1280 Optiboot (1024 bytes)
- Atmega1284 Optiboot (1024 bytes)
- Atmega2560 with fixes for watchdog timer problem (8192 bytes)
- Atmega16U2 - the bootloader on the USB interface chip of the Uno

You can use that to install or update bootloaders on the above chips (using another Arduino as the programmer).

The bootloader code is built into the sketch, so it is self-contained (it does not require an SD card, PC or anything like that).

Example of use:

```
Atmega chip programmer.
Written by Nick Gammon.
Version 1.25
Compiled on Nov  4 2014 at 07:33:18
Entered programming mode OK.
Signature = 0x1E 0x95 0x0F
Processor = ATmega328P
Flash memory size = 32768 bytes.
LFuse = 0xFF
HFuse = 0xDE
EFuse = 0xFD
Lock byte = 0xCF
Clock calibration = 0x8D
Bootloader address = 0x7E00
Bootloader length = 512 bytes.
Type 'L' to use Lilypad (8 MHz) loader, or 'U' for Uno (16 MHz) loader ...
```

Atmega_Hex_Uploader

See forum post: http://www.gammon.com.au/forum/?id=11638

This lets you:

- Verify flash memory
- Read from flash and save to disk
- Read from disk and flash a chip
- Check fuses
- Update fuses
- Erase flash memory

Most operations (except changing fuses) require an external SD card, described in the forum post. You can easily connect one by obtaining a Micro SD "breakout" board for around $US 15.

The SD card uses the hardware SPI pins, and thus the programming of the target chip uses bit-banged SPI, which means that the connections to the board to be programmed differs from the above sketches.

Example of use:

```
Attempting to enter programming mode ...
Entered programming mode OK.
Signature = 0x1E 0x95 0x0F
Processor = ATmega328P
Flash memory size = 32768 bytes.
LFuse = 0xFF
HFuse = 0xDE
EFuse = 0xFD
Lock byte = 0xCF
Clock calibration = 0x8D
Actions:
 [E] erase flash
 [F] modify fuses
 [L] list directory
 [R] read from flash (save to disk)
```

```
  [V] verify flash (compare to disk)
  [W] write to flash (read from disk)
 Enter action:
 Programming mode off.
```

Wiring for the Atmega_Hex_Uploader sketch:

```
 Arduino    SD Card
 --------------------------------
 SS         CS (chip select)
 MOSI       DI (data in)
 MISO       DO (data out)
 SCK        CLK (clock)
 +5V        5V
 Gnd        Gnd

 Arduino    Target chip/board
 -------------------------------------
 D6         MISO
 D7         MOSI
 D4         SCK
 D5         Reset
 D9         Clock of target (if required)
 +5V        5V
 Gnd        Gnd
```

This sketch uses "bit banged" SPI which is why it uses pins D4, D5, D6, D7 instead of the hardware SPI pins.