

## Abstract

# AVR Flash Loader

### **Abstract**

This project (001153) uses a WIZnet WIZ810MJ board to connect an Atmel Butterfly development board to a PC host application via ethernet. The Butterfly is based on an 8MHz ATmega169 MCU with 16K flash, 2K SRAM, 512 bytes EEPROM, 4MB dataflash, 6 character LCD, joystick, and other peripherals. The embedded part of the code for this project exists within the Butterfly as a protected bootloader, independent of any resident application.

The host server code, running in the PC, can function in an automatic or manual mode. In the automatic mode, when the Butterfly client connects to the server, its flash and/or EEPROM memories are programmed without user intervention. In manual mode the user may read or write the Butterfly's flash and/or EEPROM memories, load and save the memory images from/to disk, and edit the contents. Verification is included in all writes.

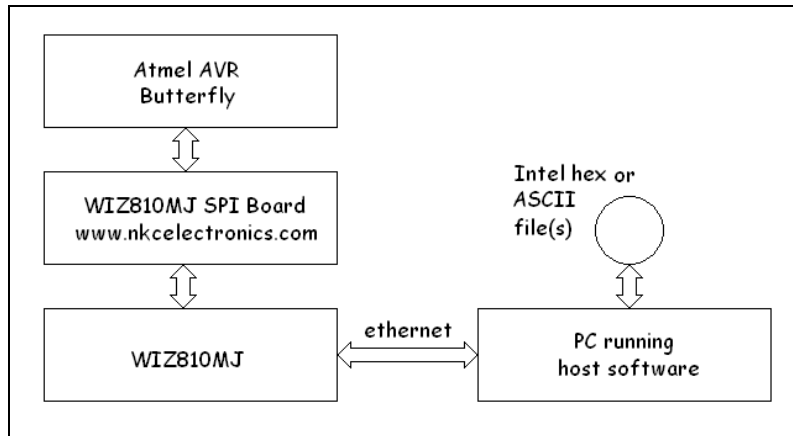
In either mode the data being programmed into the Butterfly may also be automatically modified (serial number sequencing, insertion of different text strings, etc.) via simple scripting commands. Thus each programmed Butterfly can be written with a different image.

This facilitates bulk, automated programming with resulting unique firmware. Once the Butterfly has been initially programmed with the bootloader using standard tools, bulk programming can take place in a production environment via the flexibility of the ethernet connection. The WIZnet board may continue to be a part of the Butterfly's application, or may be removed except during programming.

The Atmel Butterfly was chosen because of its low price (under \$25) and complement of existing support circuitry, thus allowing the entire project to be constructed from off the shelf components and a wiring harness. However, because of design consistency across the AVR product line, the project could easily be ported to the majority of the AVR microcontrollers with very minimal changes to the software.

### **Block Diagram**

Because of the slightly unusual spacing of the header pins (2 mm) on the WIZ810MJ, an inexpensive breakout board was purchased from [www.nkcelectronics.com](http://www.nkcelectronics.com). This board, fully populated, also provided a reset switch, power LED, and LEDs for the 5 status signals, all with pullup resistors. The SPI and other signals are brought out to a more convenient 0.1" header.



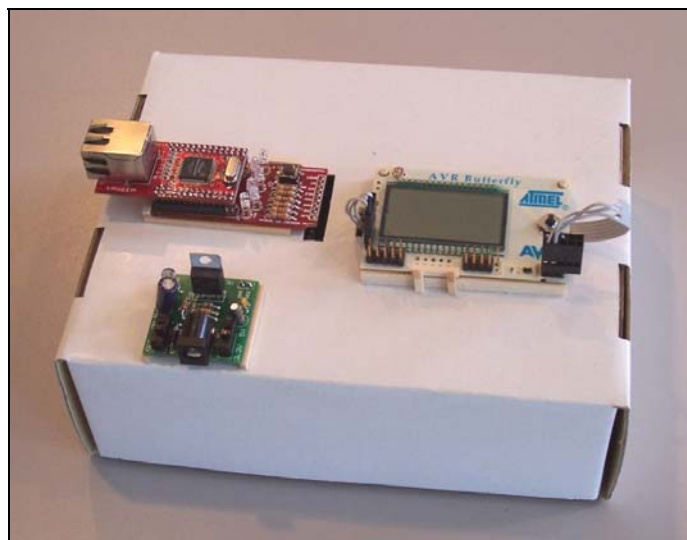
Block diagram

## Photograph

The photograph shows the commercial components luxuriously mounted to prevent electrical shorts and facilitate handling. The wiring harness extends below the visible surface.

The 6 pin header in the Butterfly's lower right corner is both its SPI and programming header (AVRs use the SPI pins for one of several methods of programming). To program the Butterfly with this bootloader application this connector is temporarily replaced with one to the programming hardware. An Atmel STK500 programmer was used, although many alternatives exist.

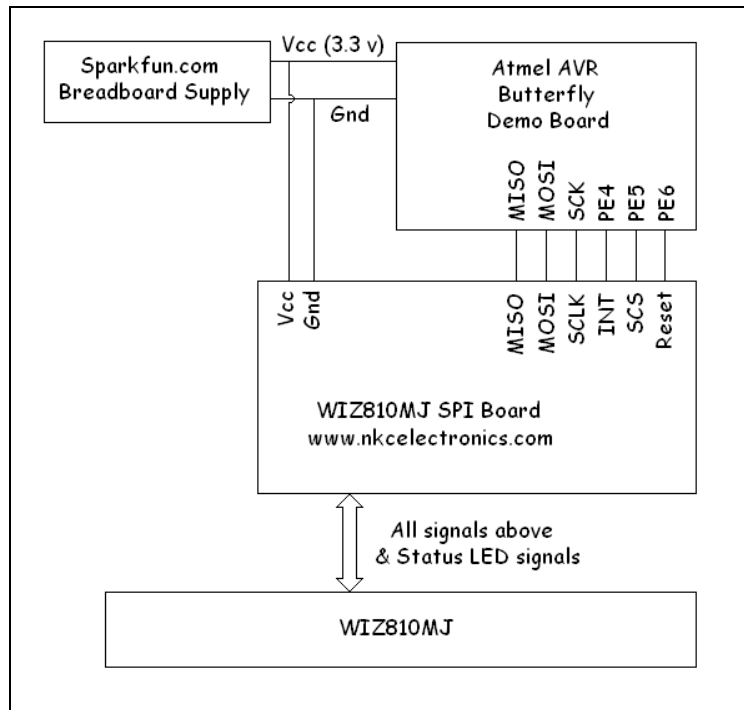
Once the bootloader has been programmed, all subsequent flash programming may be done via the ethernet connection.



Yet another use for recycled cardboard

Clockwise, from upper left: WIZ810MJ (on top) plugged into the breakout/carrier (bottom) board, the Atmel AVR Butterfly, and the Sparkfun Breadboard Power Supply.

## Schematic



## Software Code Example

Assembly language was used in the AVR because the author is a masochist. All code is original. Visual Basic 6 was used for the PC host server software.

```

; -----
;      init_WIZnet - set up the TCP stuff
;
;      highjacked MAC address: 00-0f-66-d5-83-34
;      IP address: 192.168.1.200
;      gateway: 192.168.1.1
;      subnet mask: 255.255.255.0
;      socket mem: 0 - 8K rx/tx, all others 1K (not used)
;      source port: 19001 (0x4a39)
;      destination MAC: 00-17-3f-9b-2e-54
;      destination IP: 192.168.1.100
;      destination port: 19002 (0x4a3a)

init_WIZnet:
    push    zh
    push    zl
    push    yh
    push    yl

```

```

push    r16

rcall   soft_reset_W5100 ; do a software reset

ldi     yh,high(W_GWR) ; gateway register address
ldi     yl,low(W_GWR)
ldi     zh,high(istr_1 << 1) ; the first initialization string
ldi     zl,low(istr_1 << 1)
ldi     r16,18 ; string length
rcall   WIZ_string ; write string to WIZ

ldi     yh,high(W_IMR) ; interrupt mask register address
ldi     yl,low(W_IMR)
ldi     zh,high(istr_2 << 1) ; the second initialization string
ldi     zl,low(istr_2 << 1)
ldi     r16,6 ; string length
rcall   WIZ_string ; write string to WIZ

ldi     yh,high(S0_PORT) ; source port, socket 0
ldi     yl,low(S0_PORT)
ldi     zh,high(istr_3 << 1) ; the third initialization string
ldi     zl,low(istr_3 << 1)
ldi     r16,16 ; string length
rcall   WIZ_string ; write string to WIZ

pop     r16
pop     yl
pop     yh
pop     zl
pop     zh

ret

; string 1 is: gateway (4), subnet (4), mac (6), IP (4), total = 18
istr_1: .dw
0xa8c0,0x0101,0xffff,0x00ff,0x0f00,0xd566,0x3483,0xa8c0,0xc801

; string 2 is: interrupt mask (1), timeout value (2), timeout count
(1),
; rx mem (1), tx mem (1), total = 6
istr_2: .dw 0x0f00,0x08a0,0x0303

; string 3 is: source port (2), dest MAC (6), dest IP (4), dest port
(2),
; max seg size (2), total 16
istr_3: .dw 0x394a,0x1700,0x9b3f,0x542e,0xa8c0,0x6401,0x3a4a,0xb405

```