Atmel AT90S1200 Fuse Bit Programmer

Tom Wheeler - DeVry / KC

June 2002

This is a project that I didn't really want to do! By accident, I ordered a sizable quantity of Atmel AT90S1200 parts in the "wrong" style. You probably know that the 1200 parts require an external crystal, and the 1200A parts run with an internal RC oscillator. Well - I needed 1200A devices *today* and had only the "1200" type parts on hand.

Atmel designed the '1200 with a programmable "fuse" register. These fuses control whether serial programming is possible (SPIEN) and whether the device runs on an external crystal or internal RC oscillator (RCEN). However, this register can't be programmed with a serial programmer - which is what I use for these parts!

Supposedly, the programmer supplied in the Atmel STK-200 starter kit can program the "fuse" bits on an AT90S1200 device. Yep, it sure can! I found out a few months ago that it insists on turning off the RCEN fuse on *all* AT90S1200 devices programmed in it. I couldn't get it to do otherwise. Therefore, all the parts programmed by this unit end up with the RCEN option disabled -- not a very fun thing to find in the midst of a project! (If someone out there has figured out how to make this unit properly program the RCEN bit, let me know!)

Well, with a large quantity of '1200 parts on hand and a deadline to meet, I decided to go for broke. Why not roll my *own* parallel programmer?

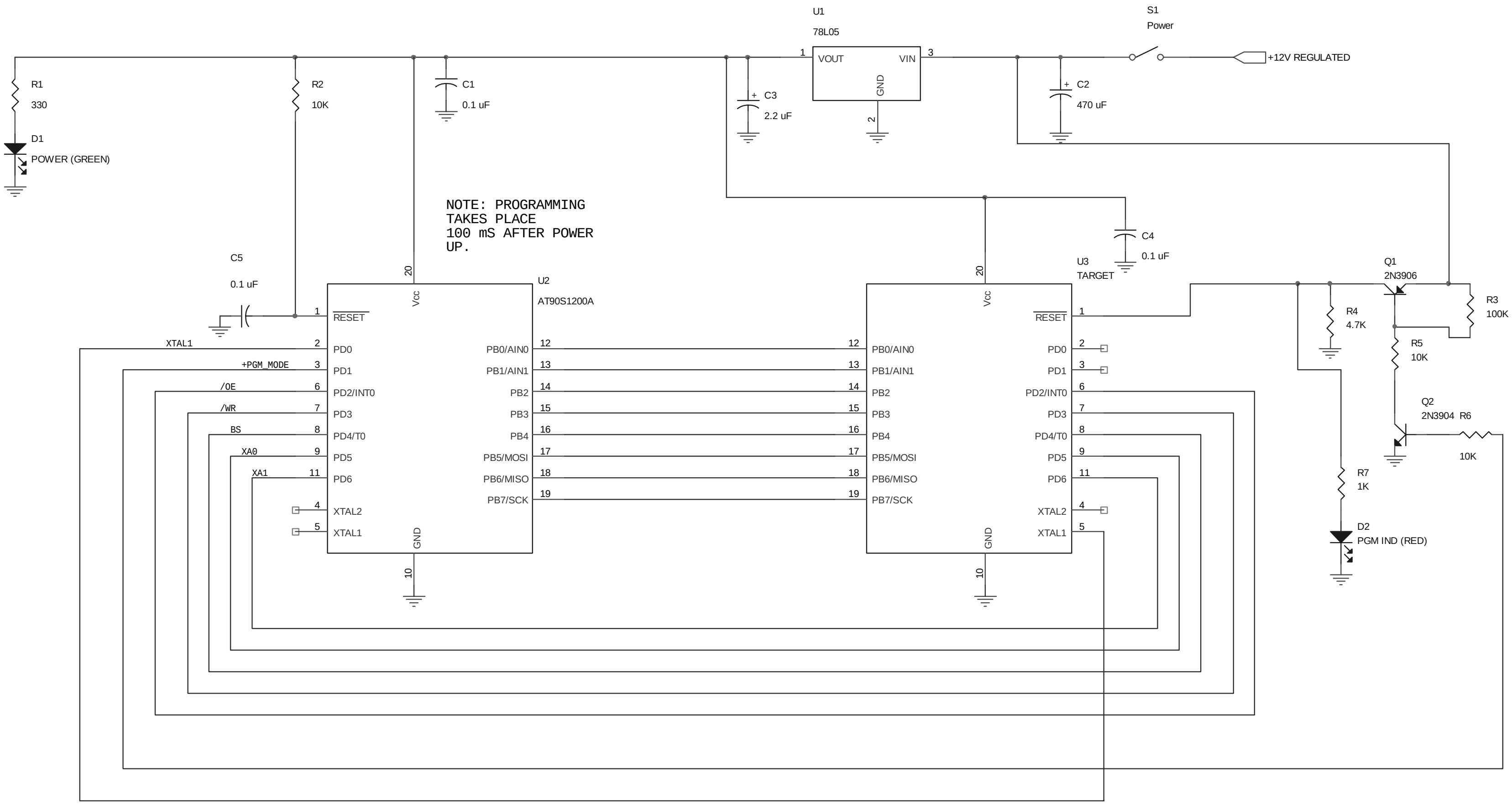
I first considered using a PC's parallel port. However, there weren't enough I/O pins to do the job without adding data latches. That sounded like work! So I instead decided to throw an AT90S1200A at the circuit.

Figure 1 shows the circuit. It has only one purpose in life -- to program the FUSE register the way I want. It isn't designed to be a full-blown programmer. It just solves the problem I ran into. The software in U1 takes care of all the work, and for simplicity, like pins on U1 and U2 (the target MCU) are just strapped together. Q1 and Q2 switch the +12V power supply to put U2 into programming mode.

When you use this device, simply insert a chip into the U2 socket, and turn on the power. The *power* LED will light, and the *programming* LED will light for a fraction of a second (it takes almost no time to program the FUSE bits.) Turn the power off, remove U2. That's all there is to it!

The MCU in U2's position will be erased after this procedure, and will have the RCEN bit programmed to "0" so that it can run on the internal RC oscillator.

The programmer doesn't verify the RCEN bit, nor does it allow the user to choose which way the bit will be programmed. Care to add these features? With a few additions, this thing has possibilities...



NOTE: PROGRAMMING
TAKES PLACE
100 mS AFTER POWER
UP.

Figure 1: Atmel AT90S1200 Fuse Bit Programmer

```
;
; pgmr.asm
;
; Circuit to program the fuse bits of an AT90S1200 to make it into an AT90S1200A (RC-osc)
; part.
;
; Target: AT90S1200A (ATMEL)
;
; Version 1.0
;
; Author: Wheeler, T (N0GSG)
;
;
;
; Register usage:
;
;


.equ    portd=$12
.equ    ddrd=$11
.equ    pind=$10
.equ    portb=$18
.equ    ddrb=$17
.equ    pinb=$16
.equ    acsr=$08
.equ    tccr0=$33
.equ    tcnt0=$32
.equ    mcucr=$35
.equ    gimsk=$3b
.equ    timsk=$39
.equ    sreg=$3f



        .org   0              ;code always begins at address 0
        rjmp   start
        rjmp   ext_int0
        rjmp   tim0_ovf
        rjmp   ana_comp

        .org   4              ;jump over IVT


;*************** main ***************

start:  rcall  initports      ;initialize ports



;****************************************************************************
; Main Program Logic:
;
; a) Wait 200 mS after POR. WR=1, RESET & BS = 0 during this time.
; b) Set BS=0, +PGM_MODE ("RESET")=12 V
; c) Wait 10 mS
; d) Set XA1,XA0 to "10" -- Command Load, BS = 0
; e) Set Data pins to 0100 0000, pulse XTAL1 to send command
;
; f) Load Data pins with 0000 0000 (D5=SPIEN, D0=RCEN)
; g) Pulse /WR low to set the config
; h) Wait 200 mS
; i) Remove the +12V supply
;
;****************************************************************************



main:   ldi    r16,0b00001100 ;D0=XTAL1=0
                              ;D1=PGM_MODE=0 (OFF)
                              ;D2=/OE = 1
```

```
                               ;D3=/WR = 1
                               ;D4=BS = 0
                               ;D5=XA0 = 0
                               ;D6=XA1 = 0
                               ;D7=0=UNUSED

        out     portd,r16
        rcall   delay          ;wait 200 mS

        in      r17,portd
        sbr     r17,2          ;PGM_MODE = 1
        out     portd,r17

        rcall   delay1

;
;0) Perform a chip erase
;
        in      r17,portd
        sbr     r17,64
        cbr     r17,32         ;XA1=1,XA0=0:Load CMD
        out     portd,r17
        ldi     r16,0b10000000 ;chip erase
        out     portb,r16
        rcall   delay1
        rcall   pulse_xtal1
        rcall   pulse_we
        rcall   delay1

;
;1) Program FUSE bits
;
        in      r17,portd
        sbr     r17,64
        cbr     r17,32         ;XA1=1,XA0=0:Load CMD
        out     portd,r17
        ldi     r16,0b01000000 ;CMD:Set Fuse Bits
        out     portb,r16
        rcall   delay1
        rcall   pulse_xtal1


;
;Set data: CONFIG Data byte
;
        in      r17,portd
        sbr     r17,32         ;XA0=1
        cbr     r17,64         ;XA1=0
        out     portd,r17      ;XA1=0,XA0=1:Load Data
        ldi     r16,0b11011110 ;CONFIG data
                               ;D5=SPIEN=0
                               ;D0=RCEN=0
                               ;D7,D6,D4-D1=1
                               ; (unprogrammed)

        out     portb,r16
        rcall   delay1
        rcall   pulse_xtal1

;
;WRITE the CONFIG data
;


        rcall   pulse_we
        rcall   delay

        ldi     r16,0b00001100 ;PGM MODE back off
        out     portd,r16

end:    rjmp    end            ;all done
```

```
pulse_we:
        in     r17,portd
        cbr    r17,8          ;WR enable
        out    portd,r17

        rcall  delay1

        in     r17,portd
        sbr    r17,8
        out    portd,r17      ;WR inactive again
        ret


pulse_xtal1:
        in     r17,portd
        sbr    r17,1          ;XTAL1=1
        out    portd,r17

        rcall  delay1

        cbr    r17,1          ;XTAL1=0
        out    portd,r17
        ret

;
;
;*** Wait for approximately 200 mS (depends on 1 MHz CPU2 clock)
;

delay:  clr    r2
        clr    r3
dl1:    dec    r2
        brne   dl1
        dec    r3
        brne   dl1
        ret


delay1: ldi    r16,10
        clr    r2
        mov    r3,r16
        rjmp   dl1


;*******************************************************
;
; EXT_INT0 handler. No action, this interrupt is disabled
; in this implementation
;
;*******************************************************

ext_int0:      reti




;*******************************************************
;
; Timer Overflow Handler. No action, disabled in this
; implementation.
;
;*******************************************************


tim0_ovf:      reti
```

```
;*******************************************************
;
; ANA_COMP interrupt handler.
;
; Action: No action, disabled in this
; implementation.
;
;*******************************************************


ana_comp:      reti




;
;******************* initports ******************
;
;Set port B as all outputs
;Set port D as all outputs
;
;R16 is destroyed.
;
;************************************************
;

initports:
        clr    r16
        out    portd,r16      ;set data to $00 for both ports
        out    portb,r16

        ser    r16
        out    ddrd,r16
        out    ddrb,r16
        ret
```