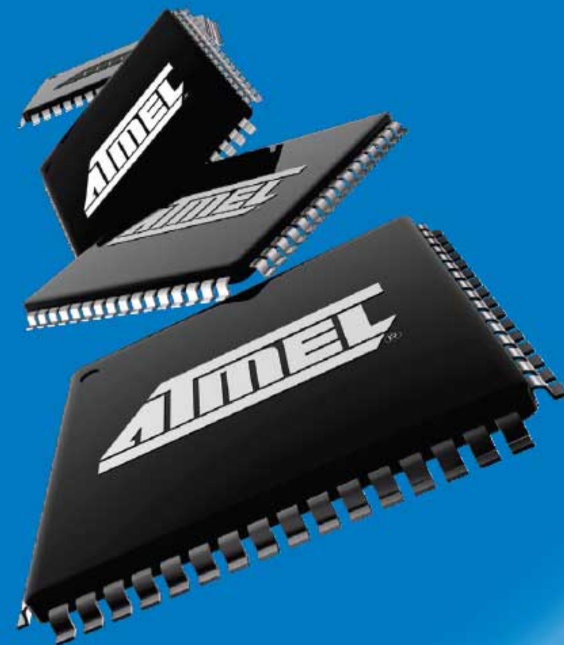
AVR®

8-bit Microcontrollers

AVR®32

32-bit Microcontrollers and Application Processors



Control Transfer

February 2009

ATMEL®

Everywhere You Are®

Design Objective

When the user presses the button, read first 3 switches (least significant), if the number is less than or equal to 5 then calculate factorial. If greater than 5 turn on decimal point. Display the least significant 4 bits of the answer.

My Design Steps

Step 1: Initialized Ports

```
; Disable interrupts and configure stack pointer for 328P
cli

; Initialize Switches with Pull-up resistors and Test LEDs
in    r16,DDRC       // input Port C Data Direction Register (0x07) for switches 5 to 0
cbr   r16,0b00111111 // define bits 5 to 0 as input (clear bit register)
out   DDRC,r16       // output

in    r16,PORTC      // input Port C Register (0x08) for switches 5 to 0
sbr   r16,0b00111111 // add pull-up resistors (PUR)
out   PORTC,r16      // output

in    r16,DDRD       // input Port D Data Direction Register (0x0A) for switches 7 to 6
cbr   r16,0b11000000 // define bits 7 to 6 as input (clear)
out   DDRD,r16       // output

in    r16,PORTD      // input Port D Register (0x0B) for switches 7 to 6
sbr   r16,0b11000000 // add pull-up resistors (PUR)
out   PORTD,r16      // output

; Initialize SPI Port and Test LEDs
in    r16,DDRB       // Input from Port B Data Direction Register (DDRB) at i/o address 0x04
sbr   r16,0b00101111 // Set PB5, PB3, PB2 (SCK, MOSI, SS) and PB1, PB0 (TEST LEDs) as outputs
out   DDRB,r16       // Output to Port B Data Direction Register (DDRB) at i/o address 0x04

in    r16,PORTB      // input Port B Register (0x05) bit 2 (SS) at i/o address 0x05
cbr   r16,0b00000111 // bit 1 (TEST LED1), bit 0 (TEST LED0)
out   PORTB,r16      // output

ldi   r16,0b01010001 // Set SPCR Enable (SPE) bit 6, Master (MSTR) bit 4,
                     // clock rate fck/16 (SPR1 = 0,SPR0 = 1)
out   SPCR,r16       // Output to SPI Control Register (SPCR) at i/o address 0x2c
```

Step 2: Turned on LED 0 to indicate initialization complete

```
    sbi     PORTB, 0        // Turn on LED 0
```

Step 3: Wrote code to pulse the clock

```
start:
    cbi     PORTD, 5
    sbi     PORTD, 5
```

Step 4: Read in pin waiting for button to be pressed (**Loop Example 1**)

```
// check button
    sbic    PIND, 2
    rjmp    start
```

Step 5: Need to filter out Bounce (**Loop Example 2**)

```
delay_50:
    ldi     r16, 0          // 256
wait:
    dec     r16             //   1 clock cycle
    brne    wait            // + 2 cycle if true, 1 cycles if false
                            //   3 cycles x 256 - 1 = 599 x 1/16 MHz = 48 usec
```

Maximum delay that could be generated was only 48 usec

Step 6: Added a NOP instruction, max delay was now 64 usec

Set delay for nice even number of 50 usec

```
delay_50:
    ldi     r16, 200        // 200 = 0xC8
wait:
    nop                     //   1 clock cycle
    dec     r16             //   1 clock cycle
    brne    wait            // + 2 cycle if true, 1 cycles if true
                            //   4 cycles x 200 - 1 = 799 x 1/16 MHz = 50 usec
```

Step 7: Made an outside loop of 10 (**Loop Example 3**)

```
delay_500:
    ldi     r17, 10
delay_50:
    ldi     r16, 200        // 200 = 0xC8
wait:
    nop                     //   1 clock cycle
    dec     r16             //   1 clock cycle
    brne    wait            // + 2 cycle if true, 1 cycles if true
                            //   4 cycles x 200 - 1 = 799 x 1/16 MHz = 50 usec
    dec     r17
```

```
    brne    delay_50       // 10 x 50 usec = 500 us (approx)
```

Step 8: Converted loop to a subroutine so I could change condition to button release.

```
; -------------------------
Delay500:
    push    r16
    push    r17

    ldi     r17, 10        // was 10
delay_50:
    ldi     r16, 200       // 200 = 0xC8
wait:
    nop                    //   1 clock cycle
    dec     r16            //   1 clock cycle
    brne    wait           // + 2 cycle if true, 1 cycles if true
                           //   4 cycles x 200 - 1 = 799 x 1/16 MHz = 50 usec
    dec     r17
    brne    delay_50       // 10 x 50 usec = 500 us (approx)

    dec     r18
    brne    delay_500      // 10 x 50 usec = 500 us (approx)

    pop     r17
    pop     r16
ret
```

Step 9: Check for button pressed and then released

```
start:
    cbi     PORTD, 5
    sbi     PORTD, 5

// check button down
    sbic    PIND, 2
    rjmp    start

    rcall   Delay500   // remove bounce
check_button:
    cbi     PORTD, 5
    sbi     PORTD, 5

// check button up
    sbis    PIND, 2
    rjmp    check_button
    rcall   Delay500   // remove bounce
```

Step 10: Read Switch and check if less than or equal to 5

```
    in      r16, PINC
    cbr     r16, 0b11110000 // clear undefined bits

    cpi     r16, 6          // no unsigned less than or equal to 5
    brlo    factorial
// error condition
    ldi     r16, 0x80       // decimal point
    mov     r8, r16
    rcall   writeDisplay
    rjmp    start
```

Step 11: Calculate Factorial (**Loop Example 4**)

```
factorial:
    ldi     r17, 1
    mov     r0, r17
calculate:
    mul     r0, r16        // r1:r0 = r0 x r16
    dec     r16
    brne    calculate
```

Step 12: Convert least significant nibble to 7-segment display (Flash Program **Indirect Addressing Mode**)

```
display_answer:
    ldi     r16, 0b00001111     // limit to least significant nibble
    and     r0, r16

    ldi     ZL,low(table<<1)    // load address of look-up
    ldi     ZH,high(table<<1)
    clr     r1
    add     ZL, r0
    adc     ZH, r1
    lpm     spi7SEG, Z

    rcall   writeDisplay
    rjmp    start

//             gfedcba     gfedcba     gfedcba     gfedcba     gfedcba     gfedcba
table: .DB  0b00111111, 0b00000110, 0b01011011, 0b01001111, 0b01100110, 0b01101101
//          0           1           2           3           4           5
       .DB  0b01111101, 0b00000111, 0b01111111, 0b01100111, 0b01110111, 0b01111100
//          6           7           8           9           A           B
       .DB  0b00111001, 0b01011110, 0b01111001, 0b01110001
//          C           D           E           F
```