CEEN 1060

Microprocessor Applications

AVR Programming Tutorial

Prepared For

The Department of Computer and Electronics Engineering

University of Nebraska-Lincoln
Peter Kiewit Institute

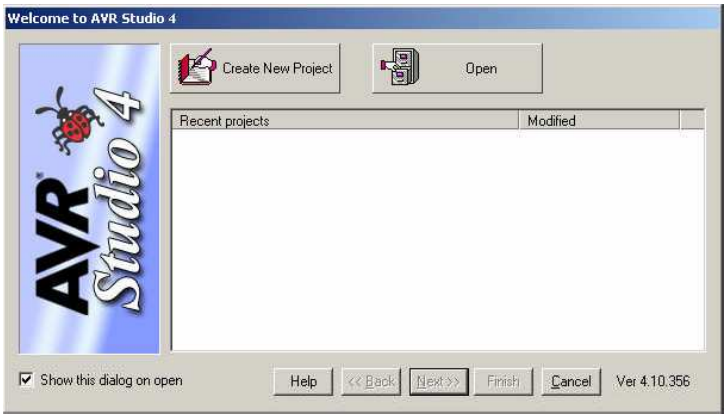
SPRING 2007

Programming the Microcontroller:

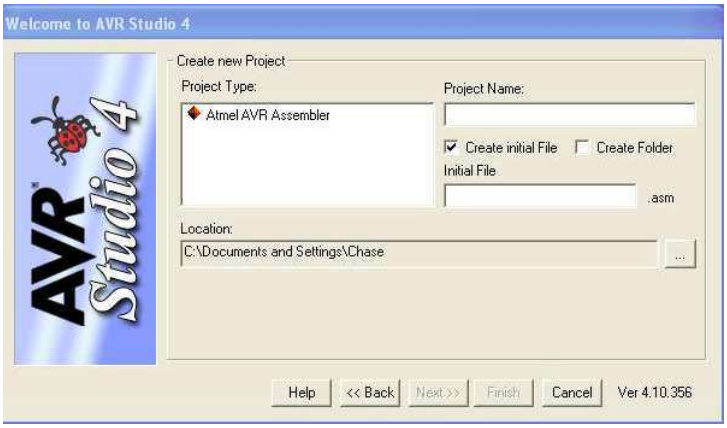
1. Make sure the ISP (in system programmer) is plugged into the back of the computer.

2. Double click AVR Studio icon on desktop to open application.

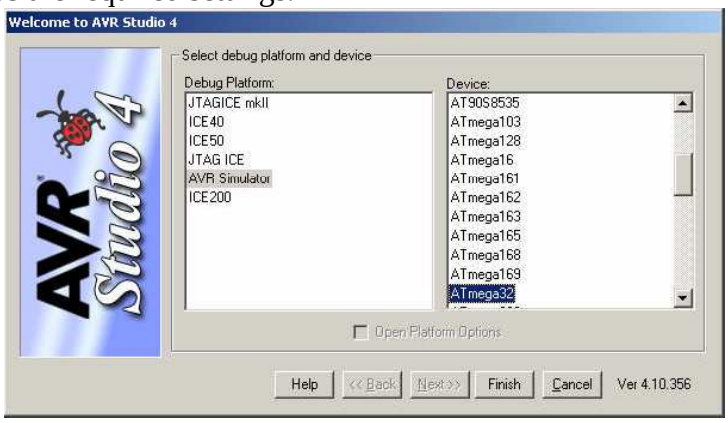
3. The following screen will appear. This allows the user to either create a new a project or open an existing project.



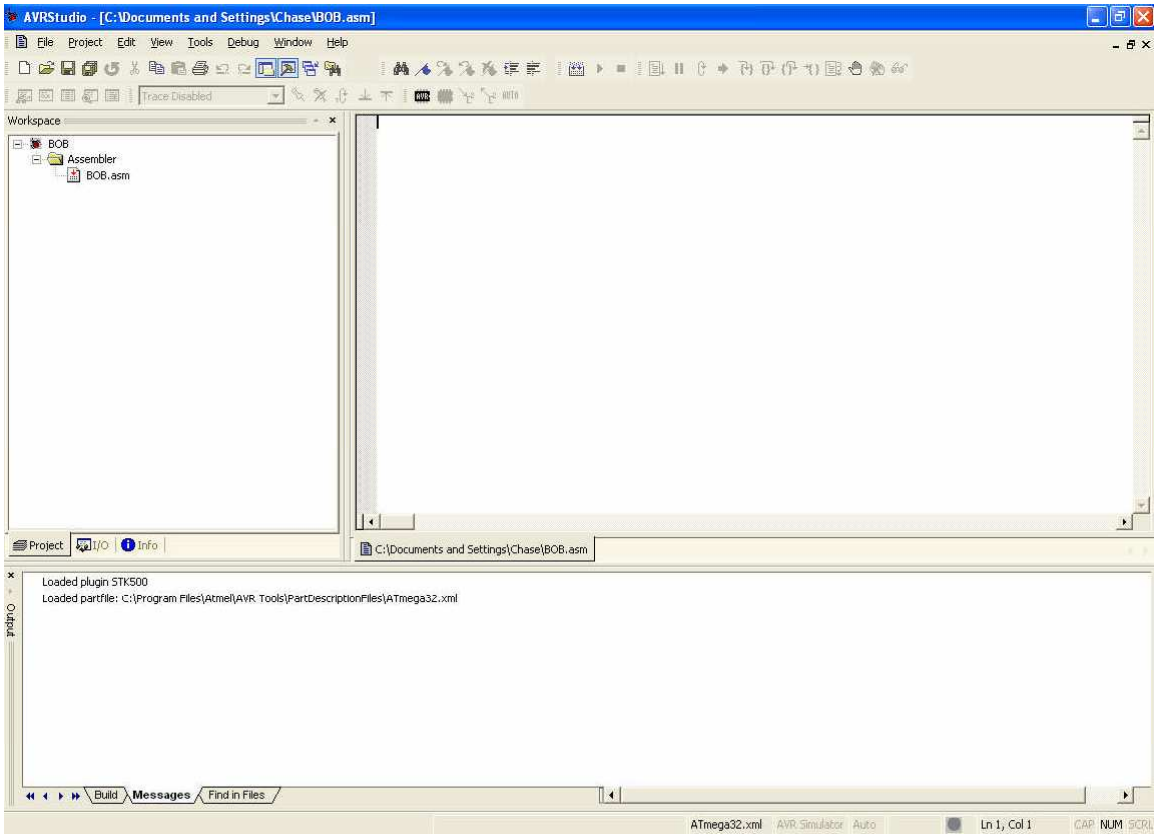
4. Since this will be your first project click “Create New Project.” The following screen will now appear. Set the “Location” and “Project Name” to values of your choosing.



Select “AVR Simulator” from “Debug Platform.” Select “ATmega32” from “Device.” The screen below shows the required settings.



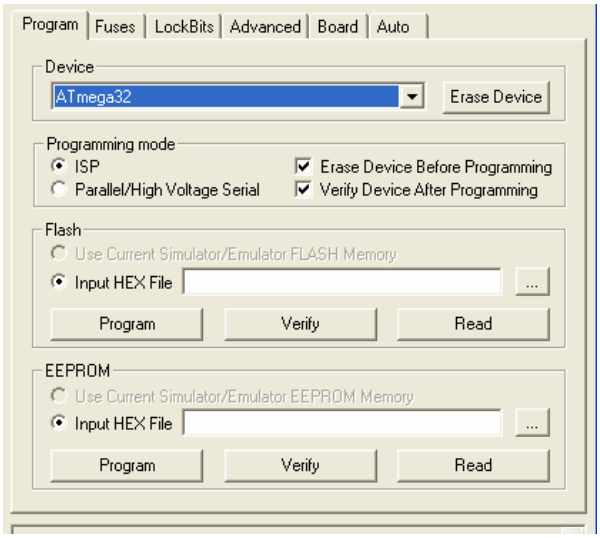
5. AVR Studio will create an assembly file for the “Project Name” chosen in Step 4. The right pane of the window is the contents of the assembly file that has been created. You will type your code here.



6. When you are complete, save your code and project. Then click “Project” and then “build.”

7. Once this is completed, you will have a hex file that can be used to program the ATmega32. If it doesn’t complete successfully you will receive errors that will be listed in the bottom pane of the AVR Studio. These are the errors for a given line of code. This is useful for tracking down errors, as it lists what line the error is on.

8. Next click “Tools” and then “AVR Program.” This brings up the following screen.



9. Verify that device is “ATmega32.” Then click the “Fuse” tab. Verify that “Enable JTAG Interface” is disabled. Verify that “Brown-out detection level=2.7 V” is enabled. Verify that “Int. RC Osc. 1 MHz, Start-up time: 6 CK + 0 ms” is selected. Lastly verify that “Boot flash section size =256 words” is selected. Everything else should not be selected.

10. Click the “Program” tab. Click the “…” icon by “Input HEX File.” Select the hex file that will be used to program the ATmega32. This HEX file will have the same name as the assembly file.

11. After the HEX file has been selected click “Program.” This will program the ATmega32. Once this is completed verify that the code works as expected.

For more information on the ATmega32, please logon to the Atmel Website for datasheets. The instruction set summary can be found on pages 327-329. The Register summary can be found on pages 325-326.

```
;-----------------------------------------------------
;-----------------------------------------------------
;CEEN 1060  Microprocessor Applications
;Computer and Electronics Engineering Department
;Peter Kiewitt Institute
;Fall 2005
;Program For ATMega32 Target Board
;-----------------------------------------------------
;-----------------------------------------------------

;-----------------------------------------------------
;-----------------------------------------------------
;This program is designed to replace the functions
;of the analog control board on the TekBot with a
;progammable digital control board.  This code uses
;interrupts to detect the bump sensor.  Depending on which
;sensor is detected, the bot backs up, turns, and then
;proceeds forward.  It also reads in the DIP switch after
;being reset and then rotates that value on the LED bar.
;A default text message is displayed on the LCD until
;the first sensor is triggered.  The LCD then displays a
;'L' or 'R' depending on which bump trigger is hit.
;-----------------------------------------------------
;-----------------------------------------------------




;-----------------------------------------------------
.include "m32def.inc"   ;Includes the ATmega32 definitions file
;-----------------------------------------------------




;-----------------------------------------------------
;-----------------------------------------------------
;Set up the assembler definitions
;This must be after the include file
;-----------------------------------------------------

;LCD definitions
.equ RS           = 64        ; RS Signal
.equ RW           = 32        ; RW Signal
.equ ENABLE = 0x10            ; ENABLE Signal

.def CLRFLAG      = R25 ;use r25 to store clrFlag value
                                ;used to know to clear screen again or
not
                                ;durring ISR1 or ISR0

.equ LCD_PORT     = PORTA     ; LCD data port
.equ LCD_DDR      = DDRA      ; LCD port data direction register
.equ LCD_PIN      = PINA      ; LCD port input pins
.equ MOT_PORT     = PORTD
.equ MOT_DDR      = DDRD
.equ MOT_PIN      = PIND
.equ LED_PORT     = PORTC
.equ LED_DDR      = DDRC
.equ LED_PIN      = PINC
```

```
.equ DIP_PORT     = PORTB
.equ DIP_DDR      = DDRB
.equ DIP_PIN      = PINB

;---------------------------------------
;    Motor control constants
;
;    Motor Pins
;           0-GND ;USED TO CONNECT TO BUMP INPUT
;           1-GND ;CAN USE FOR OTHER PURPOSE AND CON BUMP TO GND DIRECTLY
;       4-Enable left
;       5-Direction left
;       6-Enable right
;       7-Direction right
;
;    LOGIC            EFFECT
;
;    Enable Low     = Motor ON
;    Enable High    = Motor OFF
;    Direction Low  = Reverses
;    Direction High = Forwards
;---------------------------------------
.equ LGND         = 0         ;LEFT GND
.equ RGND         = 1         ;RIGHT GND
.equ LEFTEN       = 4   ; Left enable pin
.equ LEFTDIR      = 5   ; Left direction pin
.equ RIGHTEN      = 6   ; Right enable pin
.equ RIGHTDIR     = 7   ; Right direction pin


;-----------------------------------------------------------------
;-----------------------------------------------------------------
;LCD Display constants
;-----------------------------------------------------------------
.equ LINE1  = 0x80            ;Go to Line 1
.equ LINE2  = 0xC0            ;Go to Line 2
.equ CLEAR  = 0x01      ;Clear display set DD RAM to first location
.equ HOME   = 0x02      ;Ruturn cursor to home location
;---------------------------------------------------------
;---------------------------------------------------------

;---------------------------------------------------------
;---------------------------------------------------------
;Set up the segments and the INT. Vector Table

.org 0x0000             ;Places the following code from address 0x0000
   rjmp  RESET          ;Take a Relative Jump to the RESET Label

.org INT0addr                ;This interrupt is for the left sensor
      rjmp ISR_INT0

.org INT1addr                ;This interrupt is for the right sensor
      rjmp ISR_INT1

;-----------------------------------------------------------------
;-----------------------------------------------------------------
;Start the main code (RESET will always flow right into main_code)
;-----------------------------------------------------------------
.org 0x0046                  ;set after IVT
RESET:                  ;Reset Label
cli                             ;disable all interrupts
```

```
;-----------------------------------------------------------------
;Set up the stack poiner

ldi r16, low(RAMEND)          ;set up the stack pointer
out SPH, r16
ldi r16, high(RAMEND)
out   SPL, r16

;-------------------------------------------------
;Set up Interrupts

ldi r16, (1<<INT0)|(1<<INT1) ;Set up the external INTs
out GIMSK, r16

;-------------------------------------------------
;Set up I\O Ports

ser r16
out LED_DDR,r16   ;Set LED PORT as output

LDI   R16,(1 << LEFTDIR | 1 << RIGHTDIR | 1 << LEFTEN | 1 << RIGHTEN | 1 <<
LGND | 1 << RGND)
out MOT_DDR,r16         ;Set MOTOR PINS as output, INT PINS AS INPUT

ldi r16,0x0C            ;set pullup for pin 2 & 3 OF PORT D (INTS)
out PORTD, r16    ;Enable the pullups on the ints

clr r16
out DIP_DDR,r16   ;Set DIP AS INPUT
ser r16
out DIP_PORT,r16        ;Enable the pullups on DIP_PORT

rcall stop_motors ;MOTORS SHOULD NOT START UNTIL AFTER FIRST INT...

;-------------------------------------------------
;Set up the LCD
;-------------------------------------------------
;Define strings to be printed

blankLine:
.DB "               $ "
start_line1:
.DB "CEEN 1060 $ "
start_line2:
.DB "J & J Design Sys$"

rcall lcd_init          ;initialize the LCD display

;write the initial LCD text to be displayed
ldi zh,high(start_line1*2)    ;Load high address
ldi zl,low(start_line1*2)     ;Load low address
call write_string

ldi r16, LINE2                      ;next, line 2
call lcd_cmd

ldi zh,high(start_line2*2) ;Load the high address
ldi zl,low(start_line2*2)  ;Load the low address
call write_string
```

```
;-------------------------------------------------

;read in the value on the dip to ini the led pattern
IN    R16,DIP_PIN
out   LED_PORT,R16

;set the clrFlag to 0 until after the first int is triggered
ldi   CLRFLAG,0

sei               ;global enable ints before start main loop:

;-----------------------------------------------------------
;-----------------------------------------------------------
;note:this loop will run forever.  This can be easily modified
;perform other functions...

main_code:

;PLACE YOUR OWN CODE HERE!

;THIS DEMO CODE TAKES INPUT FROM DIP SWITCH AND USES IT AS INITIAL
;PATTERN FOR LED.  THAT PATTERN IS THEN ROTATED.

      LSL         R16                     ;SHIFT BITS ONE LEFT
      BRCC  SKIPLSB                ;IF MSB WAS NOT SET LEAVE LSB ZERO
      ORI         R16,1            ;SET LSB, LEAVE OTHER BITS AS IS
SKIPLSB:
      OUT         LED_PORT,R16
      PUSH  R16
      LDI         R16,0xFF
      RCALL DELAY
      POP         R16

jmp main_code

;-------------------------------------------------
;-------------------------------------------------
;Motor Functions
;-------------------------------------------------
;-------------------------------------------------
;Start
start_motors:
SBI   MOT_PORT,RIGHTDIR
CBI   MOT_PORT,RIGHTEN
SBI   MOT_PORT,LEFTDIR
CBI   MOT_PORT,LEFTEN
ret

;-------------------------------------------------
;Stop
stop_motors:
SBI   MOT_PORT,RIGHTEN
SBI   MOT_PORT,LEFTEN
ret

;-------------------------------------------------
;Go forward left
left_motor_forward:
SBI   MOT_PORT,LEFTDIR
CBI   MOT_PORT,LEFTEN
ret
```

```
;-----------------------------------------------
;Go forward right
right_motor_forward:
SBI   MOT_PORT,RIGHTDIR
CBI   MOT_PORT,RIGHTEN
ret

;-----------------------------------------------
;Go backward left
left_motor_back:
CBI   MOT_PORT,LEFTDIR
CBI   MOT_PORT,LEFTEN
ret

;-----------------------------------------------
;Go backward right
right_motor_back:

CBI   MOT_PORT,RIGHTDIR
CBI   MOT_PORT,RIGHTEN
ret

;---------------------------------------
;stop right motor
right_motor_stop:
SBI   MOT_PORT,RIGHTEN
ret

;---------------------------------------
;stop left motor
left_motor_stop:
SBI   MOT_PORT,LEFTEN
ret

;-----------------------------------------------
;-----------------------------------------------
;ISRs
;-----------------------------------------------



;-----------------------------------------------
;-----------------------------------------------
;left sensor interrupt
;-----------------------------------------------

ISR_INT0:
      cli                     ;disable all interrupts
      push  r16

      in          r16,SREG          ;read in value from SREG and store
      push  r16

      LDI         R16,0xF0          ;output Left pattern to led's
      OUT         LED_PORT,R16

      SBRS  CLRFLAG,0         ;IF CLEAR FLAG IS SET DO NOT CLEARSCREEN
      rcall CLEAR_LCD
      ORI         CLRFLAG,1         ;SET CLRFLAG
```

```
      ldi   r16, 0x4C         ;write "L" to LCD
      rcall       lcd_char

      rcall       left_motor_back;  ;both motors back
      rcall       right_motor_back;

      ldi   r16, 0xFF         ;delay for motors moving backwards
      rcall       delay
      ldi   r16, 0xFF
      rcall       delay

      rcall       right_motor_forward       ;turning the TekBot

      ldi   r16, 0xFF         ;delay for turn
      rcall       delay
      ldi   r16, 0xFF
      rcall       delay

      rcall       start_motors ;both motors forward again

      ;End the ISR
      ;Now, must restore original values in reg:

      pop   r16
      out   SREG,r16    ;restore SREG

      pop   r16               ;restore original value in r16
      sei                     ;enable interrupts
reti              ;end of left sensor interrupt

;-----------------------------------------------
;-----------------------------------------------
;right sensor interrupt
;-----------------------------------------------

ISR_INT1:
      cli                     ;disable interrupts
      push  r16         ;preserve value in r16

      in          r16,SREG          ;read in value from SREG and store
      push  r16

      LDI         R16,0x0F          ;output Right pattern to LEDs
      OUT         LED_PORT,R16

      SBRS  CLRFLAG,0         ;IF CLEAR FLAG IS SET DO NOT CLEARSCREEN
      rcall CLEAR_LCD
      ORI         CLRFLAG,1         ;SET CLRFLAG

      ldi   r16, 0x52         ;write "R" on LCD
      rcall       lcd_char

      rcall       left_motor_back;  ;Reverse both motors
      rcall       right_motor_back;

      ldi   r16, 0xFF         ;delay for reverse
      rcall       delay
      ldi   r16, 0xFF
      rcall       delay

      rcall       left_motor_forward    ;turn
```

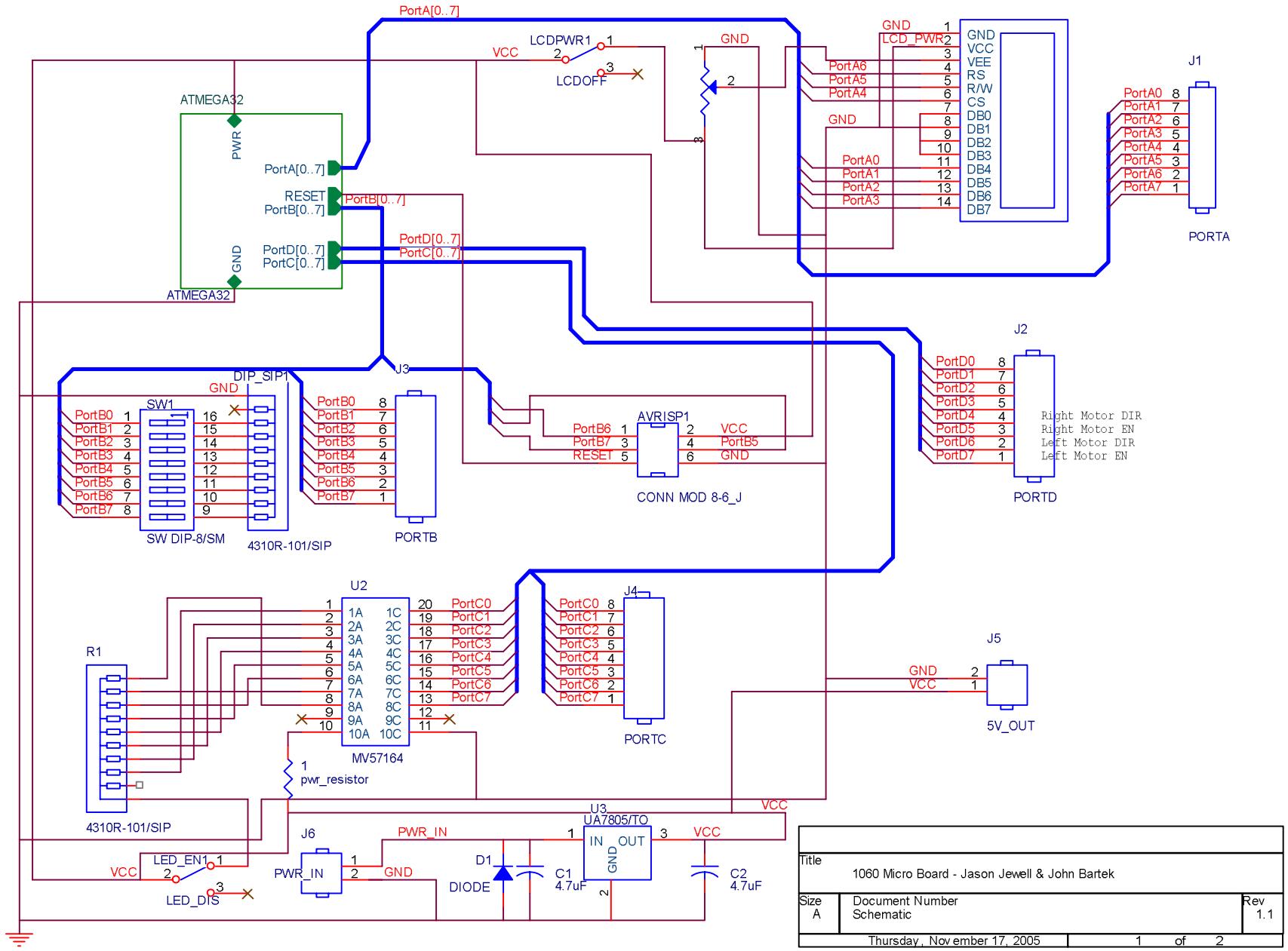
```
      ldi   r16, 0xFF              ;delay for turn
      rcall       delay
      ldi   r16, 0xFF
      rcall       delay

      rcall       start_motors           ;Both motors forward again

      ;End the ISR
      ;now, restore registers and return

      pop   r16
      out   SREG,r16    ;restore SREG register

      pop   r16               ;restore original value in r16
      sei               ;enable interrupts
reti              ;return from sensor interrupt


;-------------------------------------------------
;-------------------------------------------------
;LCD Functions
;-------------------------------------------------

;-------------------------------------------------
;Print a string to the screen
;-------------------------------------------------

write_string:

      lpm                           ;loads z into r0
      mov   r24, r0

label2:

      mov   r16, r0
      call  lcd_char

      adiw  zl,1        ;increment the character count
      lpm                     ;get the next character in r0
      mov   r16,r0
      cpi   r16, '$'    ;comparison
      brne  label2            ;branch if not equal

ret

;-------------------------------------------------
;Send a command to the LCD (in r16)
;-------------------------------------------------

lcd_cmd2:

      cbi         LCD_PORT , 6 ;Select instruction registers (RS)
      cbi         LCD_PORT , 5 ;Setect a write opperation (RW)

      out   LCD_PORT, r16                ;output to LCD
      rcall       trigger_lcd

      ldi   r17, 0x00
      out   LCD_PORT, r17                ;output to LCD
```

```
ret

;-------------------------------------
;Send a command to the LCD (in r16)
;-------------------------------------

lcd_cmd:

      push  r16         ;need to push twice
      push  r16

      ror   r16         ;rotate 4 times
      ror   r16
      ror   r16
      ror   r16

      andi  r16, 0x0F               ;and rotated r16 with 0x0F

      out   LCD_PORT, r16                 ;output to LCD

      cbi     LCD_PORT , 6       ;Select instruction registers (RS)
      cbi     LCD_PORT , 5       ;Setect a write opperation (RW)

      rcall       trigger_lcd

      ldi   r17, 0x00
      out   LCD_PORT, r17

      pop   r16                                 ;restore r16

      andi  r16, 0x0F

      out   LCD_PORT, r16

      cbi     LCD_PORT , 6       ;Select instruction registers (RS)
      cbi     LCD_PORT , 5       ;Setect a write opperation (RW)

      rcall       trigger_lcd

      ldi   r17, 0x00
      out   LCD_PORT, r17

      pop   r16                                 ;again, restore r16 to
original

ret
;-----------------------------------------------------------
;Send a character to the LCD (in r16)
;-----------------------------------------------------------

lcd_char:

      push  r16         ;need to do this twice!
      push  r16

      ror   r16               ;rotate 4 times
      ror   r16
      ror   r16
      ror   r16

      andi  r16, 0x0F   ;and rotated r16 with 0x0F
```

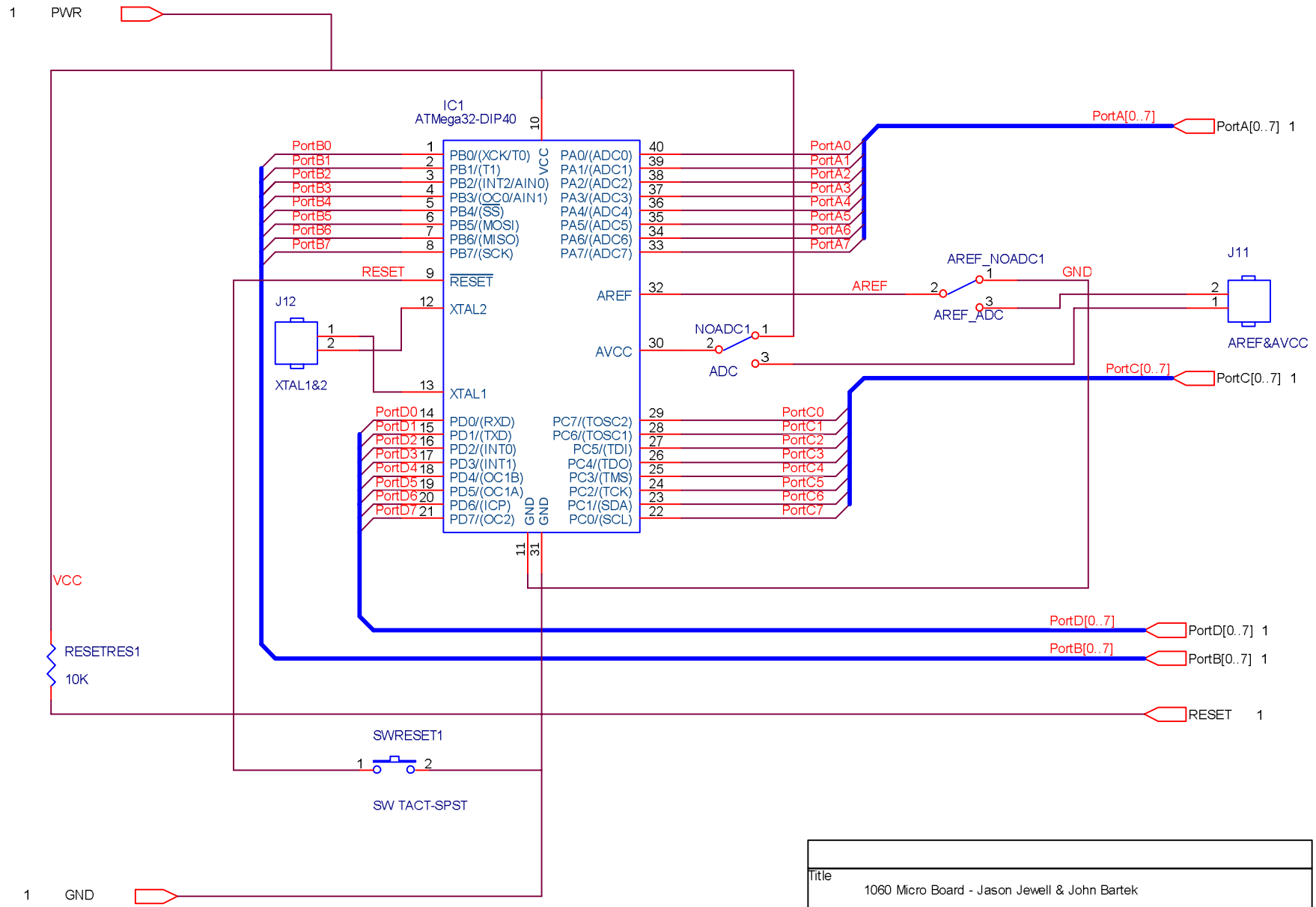
```
	out   LCD_PORT, r16                ;output to LCD

	sbi     LCD_PORT , 6       ;Select instruction registers (RS)
	cbi     LCD_PORT , 5       ;Setect a write opperation (RW)

	rcall trigger_lcd

	ldi   r17, 0x00
	out   LCD_PORT, r17                ;output to LCD

	pop   r16                                ;restore r16

	andi  r16, 0x0F                    ;and original r16 with 0x0F

	out   LCD_PORT, r16                ;output to LCD

	sbi     LCD_PORT , 6       ;Select instruction registers (RS)
	cbi     LCD_PORT , 5       ;Setect a write opperation (RW)

	rcall trigger_lcd

	ldi r17, 0x00
	out LCD_PORT, r17             ;output to LCD

	pop r16                                  ;restore r16 to original

ret
;-----------------------------------------------------
;Initialize the LCD
;-----------------------------------------------------

lcd_init:

                           ;Set the data direction pins
	ser   r16
	out   LCD_DDR, r16

	clr   r16
	out   LCD_PORT, r16     ;output to LCD

                           ;Delay
	ldi   r16, 0xFF
	rcall       delay

                           ;Start the initialization
                           ;Send 3 (once)
	ldi   r16, 0x03
	out   LCD_PORT, r16     ;output to LCD
	rcall       trigger_lcd

                           ;Delay
	ldi   r16,0xFF
	rcall       delay

                           ;Send 3 (twice)
	ldi   r16, 0x03
	out   LCD_PORT, r16     ;output to LCD
	rcall       trigger_lcd
                           ;Delay
	ldi   r16, 0xE0
```

```
      rcall       delay

                              ;Send 3 (three times)
      ldi   r16, 0x03
      out   LCD_PORT, r16     ;output to LCD
      rcall       trigger_lcd

                              ;Delay
      ldi   r16, 0xE0
      rcall       delay

                              ;Set up 4 bit data width
      ldi   r16, 0x02
      out   LCD_PORT, r16     ;output to LCD
      rcall       trigger_lcd

                              ;Delay
      ldi   r16,0x80
      rcall       delay

                              ;2 lines, 5x8 display mode
                              ;send 28h
      ldi   r16, 0x28
      rcall       lcd_cmd

                              ;send 06h
      ldi   r16, 0x06
      rcall       lcd_cmd

                              ;send 0Fh
      ldi   r16, 0x0F
      rcall       lcd_cmd

                              ;send 01h
      ldi   r16, 0x01
      rcall       lcd_cmd

                              ;send 80h
      ldi   r16, 0x80
      rcall       lcd_cmd

      ldi   r16, 0x00         ;clear r16
      out   LCD_PORT, r16

                              ;Delay
      ldi   r16, 0x80
      rcall       delay

ret

;-----------------------------------------------
;trigger the lcd to read
;-----------------------------------------------
trigger_lcd:

      push  r16          ;preserve r16
      sbi   LCD_PORT, 4 ;enable bit and delay
      ldi   r16,0x05
      rcall       delay

      cbi   LCD_PORT, 4 ;disable bit and delay
```

```
      ldi   r16,0x05
      rcall       delay

      pop   r16          ;restore register

ret

;-----------------------------------------
;clear the LCD and set curser to top left
;-----------------------------------------
CLEAR_LCD:
      ldi   r16, 0x01   ;clear lcd cmd
      rcall       lcd_cmd
ret

;-------------------------------------
;delay for a set time (in r16)
;-------------------------------------
delay:

      push  r16         ;preserve registers
      push  r18

      clr   r18         ;clear r18 register

delay_jmp:              ;conditional loop that will
      dec   r18         ;finish when r16 and r18 are equal
      brne  delay_jmp
      dec   r16
      brne  delay_jmp

      pop   r18         ;restore registers before returning
      pop   r16

ret

;-------------------------------------------------------------
;-------------------------------------------------------------
;END OF CODE
;-------------------------------------------------------------
;-------------------------------------------------------------
```



Title	1060 Micro Board - Jason Jewell & John Bartek	
Size A	Document Number Schematic	Rev 1.1
Thursday, November 17, 2005	1 of 2	


1 PWR
IC1
ATMega32-DIP40
PB0/(XCK/T0)
PB1/(T1)
PB2/(INT2/AIN0)
PB3/(OC0/AIN1)
PB4/(SS)
PB5/(MOSI)
PB6/(MISO)
PB7/(SCK)
RESET
XTAL2
XTAL1
PD0/(RXD)
PD1/(TXD)
PD2/(INT0)
PD3/(INT1)
PD4/(OC1B)
PD5/(OC1A)
PD6/(ICP)
PD7/(OC2)
PA0/(ADC0)
PA1/(ADC1)
PA2/(ADC2)
PA3/(ADC3)
PA4/(ADC4)
PA5/(ADC5)
PA6/(ADC6)
PA7/(ADC7)
AREF
AVCC
PC7/(TOSC2)
PC6/(TOSC1)
PC5/(TDI)
PC4/(TDO)
PC3/(TMS)
PC2/(TCK)
PC1/(SDA)
PC0/(SCL)
VCC
GND
PortA[0..7]
PortB[0..7]
PortC[0..7]
PortD[0..7]
J12
XTAL1&2
NOADC1
ADC
AREF_NOADC1
AREF_ADC
J11
AREF&AVCC
RESETRES1
10K
SWRESET1
SW TACT-SPST
1 GND
Title
1060 Micro Board - Jason Jewell & John Bartek
Size A
Document Number MicroController
Rev 2
Thursday, November 17, 2005
2 of 2