Summer 2004 EEL 5666

FUPA

The Garbage Can Kicking-Over Robot

Jeff Cohen
7/30/04

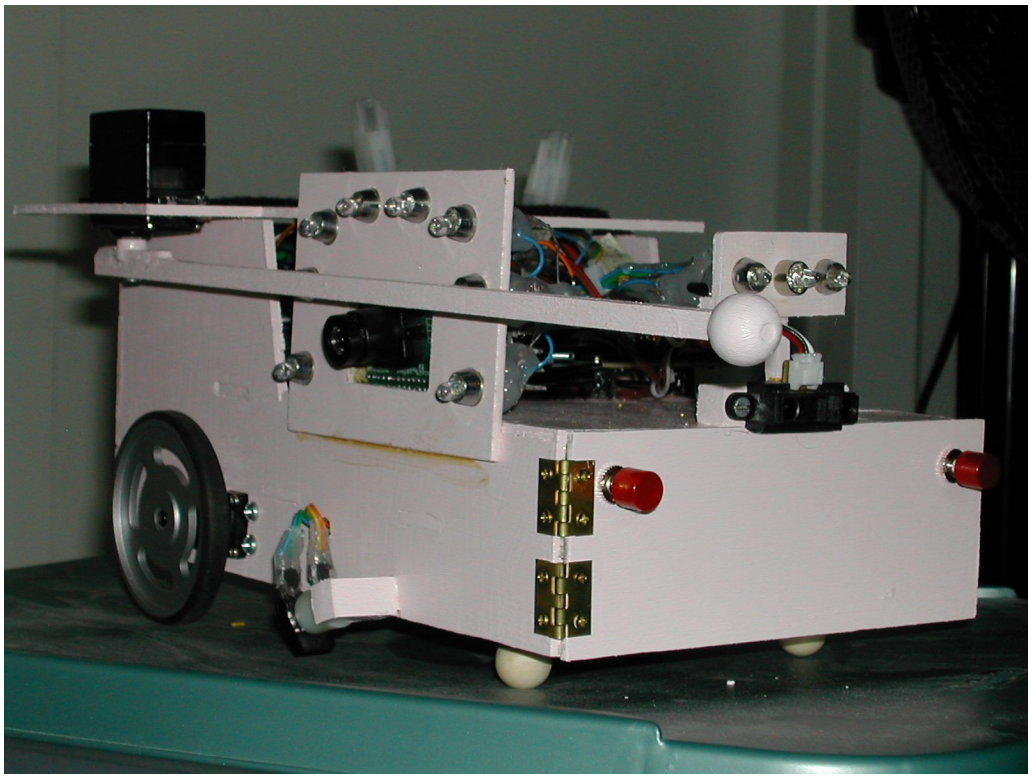


Table of Contents

Abstract

FUPA is an autonomous robot. FUPA simulates a fifth grader biking home from school on Fridays which is both trash and recycling day. FUPA will follow the sidewalk and be on the lookout for empty cans along the way. When a can is found, the color will be analyzed. If there is a red recycling bin, FUPA will knock over the can with an arm that swings out from the side. FUPA will then move on to the next house, and this cycle will continue for ever. When a can is found, the correct colored LED will also illuminate on an LED bar. FUPA will also have an obstacle destruction feature from an IR sensor, and obstacle detection from two bump switches.

Executive Summary

FUPA is a 2-WD autonomous robot. It is based on an STK500 board with an ATMega32 for its processor. FUPA follows a high contrast line using two Hamamatsu P5587 IR detectors. When a third Hamamatsu P5587 IR detector detects a line to the right of the track, a CMU cam takes a picture. If there is a red can there, FUPA will swing its arm out, and knock it over. Also, a red LED will illuminate. However, if the can was blue or green, the arm will not swing out, but the corresponding colored LED will light up.

If along the way, FUPA should detect an obstacle ahead (seen by a Sharp GP2D12 IR sensor), then FUPA enters 'destroy mode'. In destroy mode, FUPA speeds off of the track, and knocks the obstacle out of the way before returning to the track at the point FUPA left off at.

If for some reason the obstacle detection does not work, FUPA is also equipped with two bump switches which when activated will cause FUPA to cease all operations. And of course, all of the different operations include LCD feedback so you can see what FUPA is currently doing.

Introduction

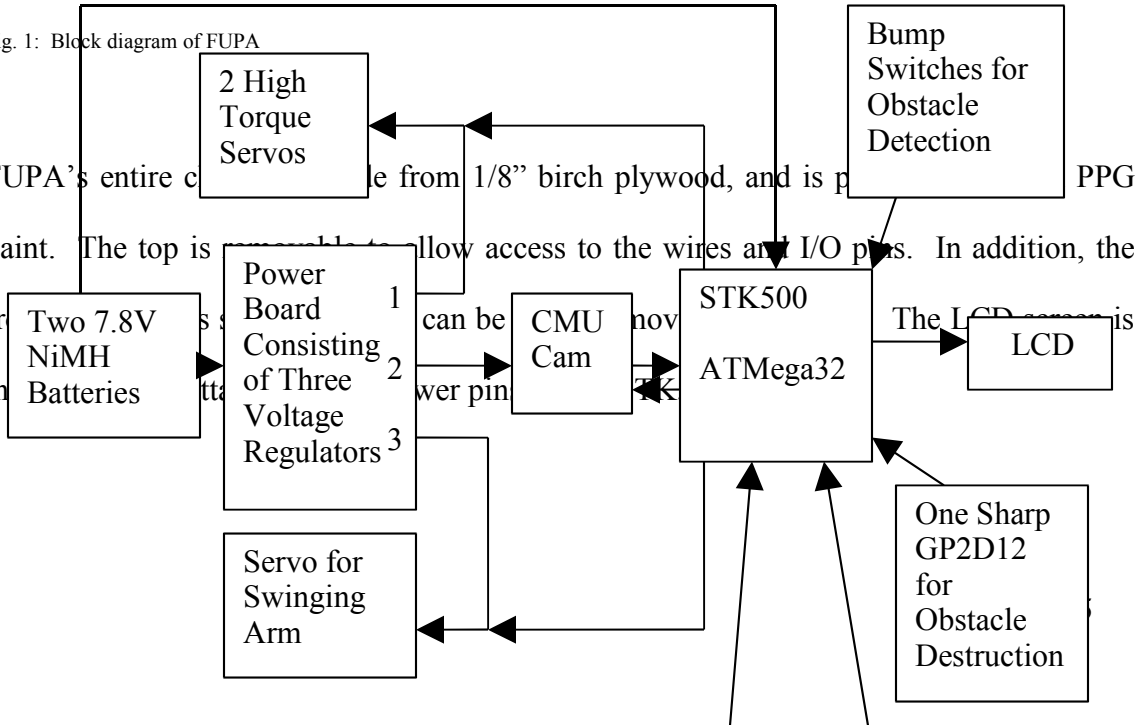
Fridays used to be the best day of the week in fifth grade. This was both garbage day and recycling day. When school would let out, a fun game was to knock over all the recycling bins as one biked home from school. This is exactly what FUPA will do. This paper describes a robot that will follow a high contrast "sidewalk" made of electrical tape. When the robot arrives at a house with garbage cans, using a CMU-CAM for color detection, it will knock over all red cans with a side-swinging arm. If any obstacle is detected in FUPA's path, FUPA will swing around and knock it out of the way.

This report will detail all of FUPA's electrical and mechanical features and characteristics. First, the platform will be presented, followed by a detailed description about each of FUPA's subsystems.

Integrated System

FUPA will have an STK500 as a platform, powered by an Atmega32 chip. Figure 1 (below) is a block diagram of FUPA's components. Figure 2 (below) is a flow chart of FUPA's software operations.

Fig. 1: Block diagram of FUPA



FUPA's entire c le from 1/8" birch plywood, and is p PPG paint. The top is removable to allow access to the wires and I/O pins. In addition, the fr s can be nov The LCD screen is th tta wer pin

One P5587 for Tab Detection

Two P5587 for Line Tracking

Fig. 2: Flow chart for FUPA's sof

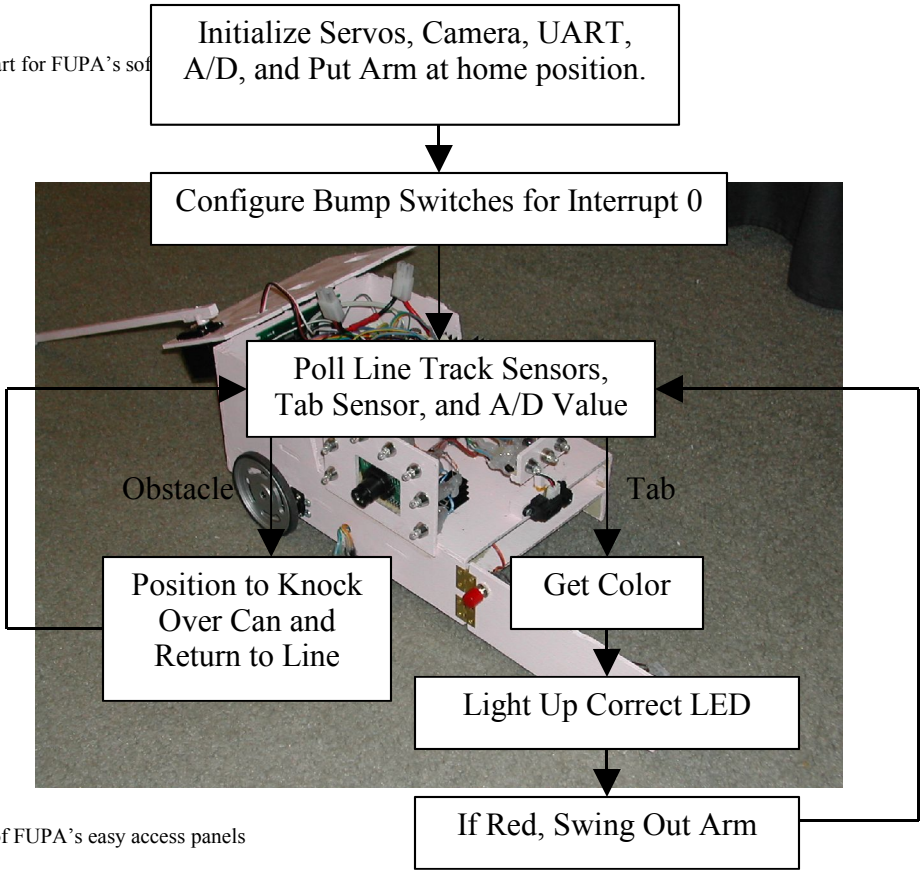


Fig. 3: Picture of FUPA's easy access panels

Power

FUPA is powered by two rechargeable 7.2V, 3300mAh, NiMH batteries. The board is powered directly from these batteries. The camera gets its power from a 7V regulator, the servos controlling the wheels get there power from a 5V regulator, and the arm servo and sensors get there power from a second 5Vregulator. See Fig. 4. All regulators were fitted with heat fins to help dissipate heat.

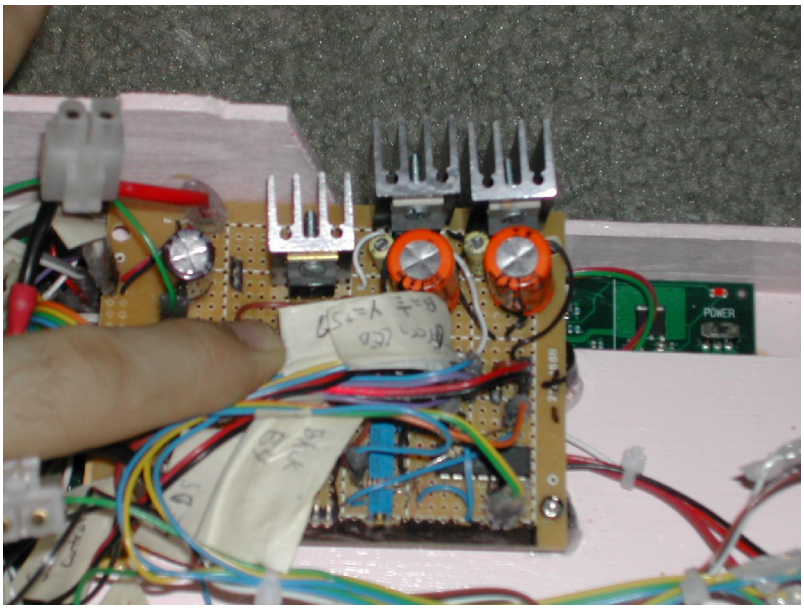
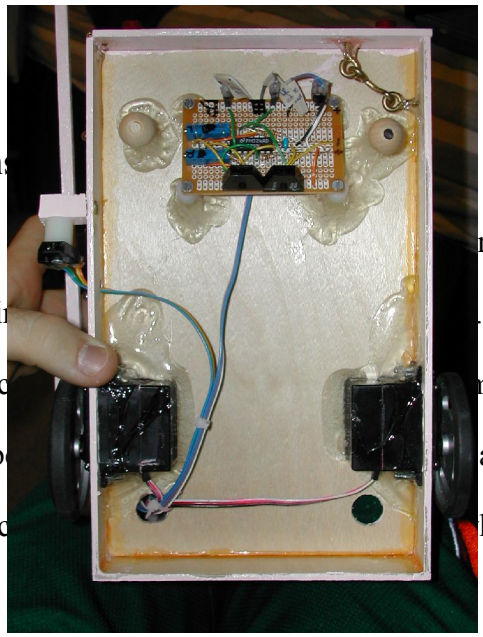


Fig. 4: Picture of FUPA's voltage regulators and heat fins

Actuation

FUPA will have two wheels which will each be powered by a hacked high-torque Futaba servo (S03 TXF 2BB). Two ¼" wood spheres are attached to a dowel rod to act as the front two stabilizers (See Fig. 5). The swinging arm will also be powered with a servo. It will use an un-hacked Futaba S03N 2BB.

Fig. 5: Picture of FUPA's servos and pseudo-wheels



Sens

Since FUPA needs to follow a sidewalk, resent a sidewalk. FUPA will follow this line usi . Tabs, noting where FUPA should analyze a can's c namatsu P5587 IR detectors. Bump switches will b a Sharp GP2D12 IR sensor will be used for obstac hat will differentiate the color of different cans.

Hamamatsu P5587 IR Detectors

Two of these sensors are used for line tracking, and one is used for tab detection. Using Will Dubel's circuit (Fig. 6), these sensors emit a 'high' value if it reads black or nothing at all. They emit a 'low' value when they are on a white surface. Through

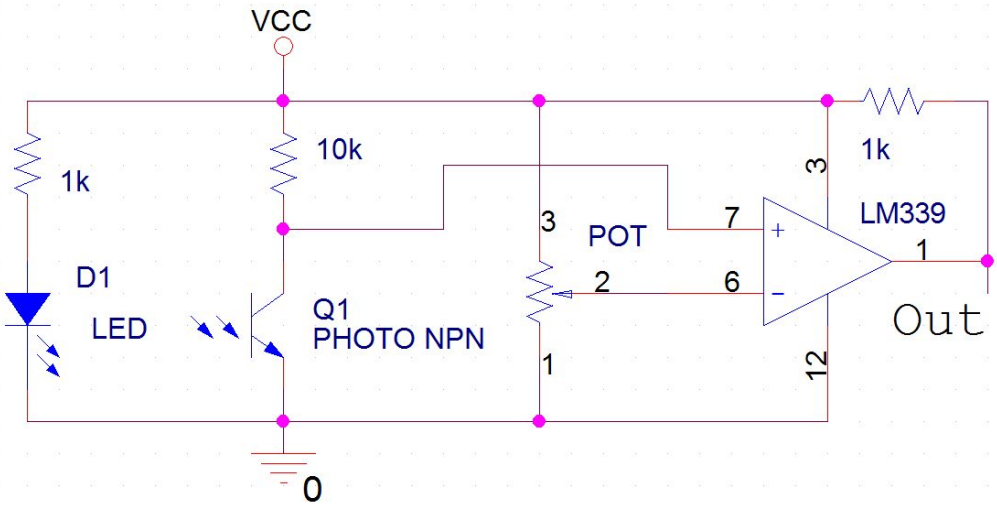


Fig. 6: Line tracking schematic

experimentation, it was determined that they work best for me 1/8" off of the ground and pointed downwards at a 40° from the horizon. FUPA's actual line tracking circuit board and tab sensor can be seen in Fig. 5.

Bump Switches

The bump switches are just two standard SPST switches. One of each switch's pins is attached to ground, and the other pin is attached to an interrupt pin. When either of these pins go 'low', the robot shuts down immediately. The bump switches can be seen in Fig. 3.

Sharp GP2D12 IR Sensor

The Sharp GP2D12 IR sensor is used for obstacle destruction, and can detect ranges from 10cm to 80cm. It is constantly returning a value to the STK500. If nothing is in FUPA's path, then the sensor will return a zero. However, when an object enters FUPA's path, the sensor returns a value which increases as the objects distance from FUPA decreases. When the sensor returns a value of $B0 or higher, FUPA will enter 'destroy mode'. This sensor can be seen above the front door in Fig. 3.

CMUcam

The CMUcam uses the UART to communicate with my ATmega32 chip at a speed of 38,400 baud. This is the fastest rate that can be used while still maintaining a 0.2% error rate. The settings of the camera changed from default mode are enabling *Polling Mode* and *Raw Data Mode*. *Polling Mode* makes the processor run faster. It makes the camera only send 1 packet of information when requested vs. a constant 17 f.p.s. *Raw Data*

Mode makes the camera transfer data bytes instead of ASCII characters. This saves the user from having to convert ASCII to useable data.

To use the CMUcam successfully, the camera requires a lot of white light. Pictures taken in my bedroom come out red. To fix this problem, ten, 16,000 m.c.d., 5 mm, white LEDs were ordered. After adding the lights, pictures were of very good quality. A sample picture can be seen in Fig. 7.

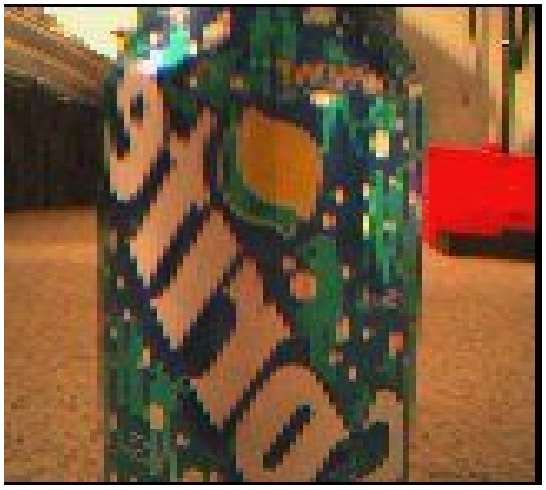


Fig. 7: A sample picture taken by the CMUcam with the white LEDs as background light

A second problem with the camera is that the picture is not centered. It is off by around 10 degrees. I had to tilt the camera when installed to account for this error. The tilted camera and the white LEDs can be seen in Fig. 3.

The voltage regulator on the camera gets very hot. To combat this problem, I installed a heat fin purchased from Radio shack.

Behaviors

Line Tracking

FUPA follows a high contrast line using two IR sensors

Destroy Mode

FUPA enters *destroy mode* when an obstacle is detected in its path. When FUPA enters this mode, the following steps are taken:

1) Turn off tab sensor

2) Move ahead for 1.5 seconds

3) Turn 45 degrees left and go straight for 0.5 seconds

4) Swing out arm

5) Reverse for 1.5 seconds

6) Turn right until the line is reacquired

7) Turn on tab sensor and continue to line track

In *destroy mode*, the obstacle is knocked out of the way, so FUPA can continue on its original path.

Camera Mode

Camera mode is entered when a tab is detected with the IR sensor. In this mode, FUPA is line tracking while snapping a picture to determine which future action to take.

Components

Quantity	Part	Price	Source
2	Wheels	6.00	Mark III
2	High Torque Futaba Servos	21.00	Mark III
1	Futaba Servo	10.50	Mark III
1	Sharp GP2D120 IR Sensor	8.50	Mark III
1	Sharp GP2D120 IR Sensor Cable	1.10	Mark III
2	Bump Switches	2.39	Radio Shack
8	White LEDs and Holders	15.00	E-Bay
3	DB9 Connectors	4.80	Radio Shack

2	Breadboards	4.00	Radio Shack
4	Heat Sinks	6.00	Radio Shack
2	Female Battery Terminals	8.00	Radio Shack
1	Variable Voltage Regulator	2.50	Radio Shack
1	Paint and Clear Coat	11.00	Lowe's
2	Hinges	2.00	Lowe's
1	Latch	3.50	Lowe's
1	Velcro	2.50	Lowe's
1	STK500	80.00	Digikey
1	ATMega32	9.00	Digikey
1	CMUcam	70.00	Used
3	Hamamatsu P5587 IRs	12.00	Digikey
1	Potentiometer	3.00	Radio Shack
1	Power Jack	5.00	Radio Shack
3	Red, Green, Blue LEDs	6.50	Radio Shack
3	Various Sized Screws	4.50	Lowe's
2	Goop	8.00	Lowe's
1	Solder	4.00	Radio Shack
2	LED Holders	2.00	Radio Shack
1	LCD Screen	8.00	Used
1	Shipping Charges	7.00	
Total		327.79	

Wire, wood, and wood glue were provided by the lab.

Conclusion

FUPA was very successful. It works as expected, and I even added the *destroy mode* feature and the LED bar. The only thing that I wish I had done different was using sonar instead of tabs/IR to determine where a can is. Overall this robot was very time consuming, eating well over 300 hours of time to get everything working 100%. But, after completion, I have never had FUPA give me a false color reading or veer off of the track

Documentations

1) Will Dubel's line tracking paper

2) Bryan Arkin's 'letter' macros

Appendices

Final Code

```
;*************************
;*   Set VTarget = 5.1V   *
;*   Set ARef = 4.9V      *
;*************************

;Jeffrey S. Cohen
;v10 with obstacle knockdown
;7/30/2004
;EEL5666C

.INCLUDE "m32def.inc"

.CSEG
;*********************
;*   Reset Vector   *
;*********************
.ORG $000
        RJMP    Reset



;*************************
;*   Interrupt 0 Vector   *
;*************************
;Bump switch interrupt
.ORG $002
        RJMP    ShutDown


;******************
;*   Constants   *
;******************
;Servo Register Renames
.EQU    RSERVO_H        =OCR1BH
.EQU    RSERVO_L        =OCR1BL
.EQU    LSERVO_H        =OCR1AH
.EQU    LSERVO_L        =OCR1AL
.EQU    ARM_SERVO       =OCR0

;Registers
.DEF    Temp            =r16            ; Temporary Reg 1
.DEF    Temp2           =r17            ; Temporary Reg 2
.DEF    Temp3           =r18            ; Temporary Reg3
.DEF    PrevIR          =r19            ; Previous IR reading
.DEF    ADCCounter      =r20            ; Counter for # of times ADC reports back a close value
.DEF    LCDReg          =r21


.DEF    Delay1          =r29            ; Delay Reg 1
.DEF    Delay2          =r30            ; Delay Reg 2
.DEF    Delay3          =r31            ; Delay Reg 3

;Ports
```

```
.EQU    LCD_PORT                =PORTC
.EQU    LCD_DDR                         =DDRC

;Constant Values
.EQU    ADC_Distance    =$B0            ; Distance where ADC reads as object too close
.EQU    Space                   =$20
.EQU    CR                              =$0D


;**************
;*   Macros   *
;**************
;Prints Letters to LCD display
.macro letter
        LDI     LCDReg,@0
        OUT     LCD_PORT,LCDReg
        SBI     LCD_PORT,5
        RCALL   Latch
        RCALL   Delay5ms
        LDI     LCDReg,@1
        OUT     LCD_PORT,LCDReg
        SBI     LCD_PORT,5
        RCALL   Latch
        RCALL   Delay5ms
.endmacro




;*************
;*   Main   *
;*************
.ORG $60
Reset:

RGB_VALUE:      .DB 0
S1:                     .DB 0
S2:                     .DB 0
R_VALUE:        .DB 0
G_VALUE:        .DB 0
B_VALUE:        .DB 0
Rdev:           .DB 0
Gdev:           .DB 0
Bdev:           .DB 0
S9:                     .DB 0

        LDI     Temp,low(RAMEND)        ; Set low stackptr
        OUT     SPL,Temp

        LDI     Temp,high(RAMEND)       ; Set high stackptr
        OUT     SPH,Temp

        RCALL   PortInit                ; Initializes port directions

        SBI     PortD,2                         ; Pull up bump switch pin

        RCALL   LCDInit                         ; Initializes LCD Screen
        RCALL   UART_Init                       ; Initialize UART
        RCALL   Camera_Init                     ; Initialize Camera
        RCALL   BufferFlush                     ; Empty out all junk data in buffer sent from camera during
initialization
        RCALL   PWMInit_Wheels          ; Initializes PWM for Wheels
        RCALL   ADC_Init                ; Initializes ADC for IR obstacle detection
```

```
        SEI                                                     ; Enable Interrupts

StartAgain:
        CBI         PortB,0                                     ; Turn off colored LEDs
        CBI         PortB,1
        CBI         PortB,4
        RCALL   PWMInit_Arm                         ; Initializes PWM for the arm servo

        RCALL   LineTrack




;*****************
;*   Port Init   *
;*****************
PortInit:
        CLR     Temp
        OUT DDRA,Temp
                                ; INPUT REGISTER
                                ; Pin0 = Sharp IR obstacle avoidance
                                ; Pin1 =
                                ; Pin2 =
                                ; Pin3 =
                                ; Pin4 =
                                ; Pin5 = IR input (Tab Detector)
                                ; Pin6 = IR input (Left)
                                ; Pin7 = IR input (Right)

        SER     Temp
        OUT DDRB,Temp
                                ; OUTPUT REGISTER
                                ; Pin0 = Red LED control pin
                                ; Pin1 = Green LED control pin
                                ; Pin2 =
                                ; Pin3 = Servo PWM (Arm)
                                ; Pin4 = Blue LED control pin
                                ; Pin5 =
                                ; Pin6 = White LED control pin
                                ; Pin7 =

        SER Temp
        OUT LCD_DDR,Temp
                                ; OUTPUT REGISTER
                                ; Pin0 = LCD DB4
                                ; Pin1 = LCD DB5
                                ; Pin2 = LCD DB6
                                ; Pin3 = LCD DB7
                                ; Pin4 = LCD enable
                                ; Pin5 = LCD register select
```

```
                                        ; Pin6 =
                                        ; Pin7 =

        LDI Temp, 0b11111010
        OUT DDRD,Temp
                                        ; Pin0 = UART input (Receive)
                                        ; Pin1 = UART output (Transmit)
                                        ; Pin2 = Bump switches (INT0)
                                        ; Pin3 =
                                        ; Pin4 = Servo PWM (Right)
                                        ; Pin5 = Servo PWM (Left)
                                        ; Pin6 =
                                        ; Pin7 =

        LDI Temp, 0b00000000            ;Configure interrupt
        OUT MCUCR,Temp

        LDI         Temp, 0b01000000    ; Enable bump switch interrupt
        OUT GICR, Temp
        RET
```

```
;*************************
;*   LCD Initialization   *
;*************************
;** PC0 = DB4 (LCD pin7)
;** PC1 = DB5 (LCD pin8)
;** PC2 = DB6 (LCD pin9)
;** PC3 = DB7 (LCD pin10)
;** PC4 = E (LCD pin6)
;** PC5 = RS (LCD pin4)
;** GND = VSS (LCD pin1)
;** GND = R/W (LCD pin5)
;** VTG = VDD (LCD pin2)

LCDInit:
;Power-On 15ms Delay
        RCALL   Delay5ms
        RCALL   Delay5ms
        RCALL   Delay5ms

;Begin 4-Bit Enable
        LDI     Temp, 3
        OUT     LCD_PORT,Temp
        RCALL   Latch
        RCALL   Delay5ms

        RCALL   Latch
        RCALL   Delay1p5ms

        RCALL   Latch
        RCALL   Delay5ms

        LDI             Temp, 2
        OUT     LCD_PORT,Temp
        RCALL   Latch
        RCALL   Delay1p5ms

;Begin 2-Line Enable
        RCALL   Latch
        RCALL   Delay1p5ms
```

```
            LDI             Temp, 8
            OUT     LCD_PORT,Temp
            RCALL   Latch
            RCALL   Delay1p5ms

;Display on, Cursor on, Blink on
            LDI             Temp, 0
            OUT     LCD_PORT,Temp
            RCALL   Latch
            RCALL   Delay1p5ms

            LDI             Temp, 15
            OUT     LCD_PORT,Temp
            RCALL   Latch
            RCALL   Delay1p5ms

;Clear screen, cursor home
            LDI             Temp, 0
            OUT     LCD_PORT,Temp
            RCALL   Latch
            RCALL   Delay1p5ms

            LDI             Temp, 1
            OUT     LCD_PORT,Temp
            RCALL   Latch
            RCALL   Delay5ms
;Initialization Complete
            RET


;*************************
;*   PWM Setup (Wheels)  *
;*************************
;
;1/8MHz = .125us per cycle
;using 64 prescaler, 1 increment of TCNT = 8us
;
;Set potentiometer so the wheel does not turn
; if the pulse is high for 1.5ms
;
;1.5ms = 187.50
;
;187.50/2=93.75
;====================================
;T=20ms, then .5T=10ms 10ms/8us=1250 ($04E2)
;
;    ^    <-- Top = 1250
;   /  \
; /      \ <-- Bottom = 0
;
;1250-93.75=1156.25 ~1156 rise
;ICR1A <-- Top
;OCR1A  <-- 1156 ($0484)
;================================================
;Full Clockwise         (Right Side)                    : 1172 0x491
;
;Neutral                                                                : 1156 0x484
;
;Full C. Clockwise      (Left Side)                     : 1141 0x477
;================================================
PWMInit_Wheels:
            LDI Temp,0b11110000
                                                        ; bits7,6    =1,1 Set up and clear down (OC1A)
                                                        ;     5,4    =1,1 Set up and clear down (OC1B)
                                                        ; bits3,2    =0,0 Force output compare OFF A and B
                                                        ; bits1,0    =0,0 PWM phase and Freq correct
            OUT TCCR1A,Temp

            LDI Temp,0b00010011         ; bit7          =0 IC noise canceler OFF
                                                        ; bit6              =0 Not used in our mode
                                                        ; bit5              =0 Reserved
                                                        ; bits4,3    =1,0 PWM phase and Freq correct
```

```
                                                ; bits2,1,0  =Clock Select Prescaler
                                                ;                                   0,1,1 : (1/8Mhz)*64=8us
        OUT TCCR1B, Temp

; Sets Top
        LDI Temp,0x04
        LDI Temp2,0xE2
        OUT ICR1H,Temp
        OUT ICR1L,Temp2

; Sets Pulse On Period
        LDI Temp,0x04
        LDI Temp2, 0x84
        OUT RSERVO_H,Temp
        OUT LSERVO_H,Temp
        OUT RSERVO_L,Temp2
        OUT LSERVO_L,Temp2

; Start TCNT's at $00
        LDI Temp, 0
        OUT TCNT1H, Temp
        OUT TCNT1L, Temp

        RET




;**********************
;*   PWM Setup (Arm)  *
;**********************
;1/8MHz = .125us per cycle
;using 256 prescaler, 1 increment of TCNT = 32us
;
;1.0ms = 31.25
;2.0ms = 62.50
;
;One period = (255+256)*32us = 16.352ms
;
;T=16.35ms, then .5T=8.175ms
;================================================
;8.175ms-0.5ms = 7.675ms
;
;7.675ms / 32us = 239.84 ~ 240 (1ms pulse = 240)
;================================================
;8.175ms-1.0ms = 7.175ms
;
;7.175ms / 32us = 224.22 ~ 224 (2ms pulse = 224)
;================================================
PWMInit_Arm:

; Sets OCR0
;219=swing out
;245=home position

        LDI Temp,245                    ; Set arm to home position on startup
        OUT ARM_SERVO,Temp

; Start TCNT's at $00
        LDI Temp, 0
        OUT TCNT0, Temp

        LDI Temp,0b01110100
                                                ; bit7                  =0   Force output compare off
                                                ; bits6,3    =1,0 CTC
                                                ; bits5,4    =1,1 Set OC0 on upcount, clear on downcount
                                                ;                                   =0,0 DISCONNECTED
                                                ; bits2,1,0  =1,0,0 Set 256 prescalar
        OUT TCCR0,Temp

        RET
```

```
;*******************
;*   ADC IR Setup  *
;*******************
ADC_Init:
        LDI Temp,0b00100000             ; bits7,6       =0,0        Use AREF as reference
                                                        ; bit5                  =1
                Left adjust
                                                        ; bits4,3,2,1,0 =0,0,0,0,0 Select ADC0

        OUT ADMUX,Temp

        LDI Temp,0b11100110             ; bit7          =1      Enable ADC
                                                        ; bit6          =1      Start conversions
                                                        ; bit5          =1      Auto trigger
enabled
                                                        ; bit4          =0      Interrupt flag (not
important)
                                                        ; bit3          =0      ADC interrupt
disable
                                                        ; bits2,1,0  =1,1,0 prescaler=64

        OUT ADCSR,Temp

        LDI     TEMP,(SFIOR&0b00011111) ; Set ADC to free running mode
        OUT     SFIOR,Temp

        CLR     ADCCounter                              ; Clear ADC counter

        RET


;*******************
;*   UART Setup   *
;*******************
UART_Init:
        LDI             Temp,12                         ; Selects baud rate (8 LSBs) 38.4k bps
        OUT             UBRRL,Temp                      ; Clock=8MHz


        LDI             Temp,0b00000000         ; Bit7 = 0                                      Must be 0
when writing to UBRRH
                                                                ; Bits6,5,4 = 0,0,0
        Reserved
                                                                ; Bits3,2,1,0 = 0,0,0,0
        Selects baud rate (4 MSBs)
```

```
        OUT     UBRRH,Temp

        LDI     Temp,0b10000110     ; Bit7 = 1      Selects proper register
                                    ; Bit6 = 0      Asynchronous mode
                                    ; Bits5,4 = 0,0 Parity disabled
                                    ; Bit3 = 0      Select 1 stop-bit
                                    ; Bits2,1 = 1,1 Select 8-bit frame
                                    ; Bit0 = 0      Clock polarity....set to
0 if asynchronous
        OUT     UCSRC,Temp


        LDI     Temp,0b00011000     ; Bit7 = 0  Rx complete interrupt disable
                                    ; Bit6 = 0  Tx complete interrupt disable
                                    ; Bit5 = 0  Data register empty interrupt
disable
                                    ; Bit4 = 1  Rx enable
                                    ; Bit3 = 1  Tx enable
                                    ; Bit2 = 0  Select 8-bit frame
                                    ; Bit1 = 0  Rx 9th bit disabled
                                    ; Bit0 = 0  Tx 9th bit disabled
        OUT     UCSRB,Temp

        RET

;*********************
;*  Camera Setup   *
;*********************
Camera_Init:
        SBI     PortB,6             ; Turn on white LEDs for picture

        RCALL   DelayS

        LDI     Temp,'R'            ; Reset camera
        RCALL   Send

        LDI     Temp,'S'
        RCALL   Send

        LDI     Temp,CR
        RCALL   Send

        RCALL   DelayL

        LDI     Temp,'P'            ; Enable polling mode
        RCALL   Send

        LDI     Temp,'M'
        RCALL   Send

        LDI     Temp,' '
        RCALL   Send

        LDI     Temp,'1'
        RCALL   Send

        LDI     Temp,CR
        RCALL   Send

        RCALL   DelayS

        LDI     Temp,'R'            ; Enable raw data output
        RCALL   Send                ; Disable 'ACK'/'NAK' responses

        LDI     Temp,'M'
        RCALL   Send

        LDI     Temp,' '
        RCALL   Send

        LDI     Temp,'3'
        RCALL   Send
```

```
        LDI     Temp,CR
        RCALL   Send

        RCALL   DelayS

        RET


;****************************************
;*  Flush out old received characters   *
;****************************************
BufferFlush:
        SBIS    UCSRA,RXC
        RET
        IN      Temp,UDR
        RJMP    BufferFlush




;**************************
;*  Get Mean RGB values  *
;**************************
GetMean:
        LDI     Temp,'G'            ; Get mean values
        RCALL   Send

        LDI     Temp,'M'
        RCALL   Send

        LDI     Temp,CR
        RCALL   Send

        RCALL   Receive1            ; 255 (decimal)
        RCALL   Receive2            ; 'S'
        RCALL   Receive3            ; Red
        RCALL   Receive4            ; Green
        RCALL   Receive5            ; Blue
        RCALL   Receive6            ; Rdev
        RCALL   Receive7            ; Gdev
        RCALL   Receive8            ; Bdev
        RCALL   Receive9            ; ':'

        LDS     Temp,Rdev
        LDS     Temp2,Gdev
        LDS     Temp3,Bdev

        CP      Temp,Temp2
        BRGE    NotGreen
        RJMP    NotRed

NotGreen:
        CP      Temp,Temp3
        BRGE    FoundRed
        RJMP    FoundBlue

NotRed:
        CP      Temp2,Temp3
        BRGE    FoundGreen
        RJMP    FoundBlue

FoundRed:
        SBI     PortB,0                 ; Turn on Red LED
        RCALL   ClearLCD
        letter 5,2                      ; load 'R'
        letter 6,5                      ; load 'e'
        letter 6,4                      ; load 'd'

        RCALL   SwingArm                ; Knock over can
```

```
        RJMP    MoveOffTab

FoundGreen:
        SBI             PortB,1                 ; Turn on Green LED

        RCALL   ClearLCD
        letter 4,7                              ; load 'G'
        letter 7,2                              ; load 'r'
        letter 6,5                              ; load 'e'
        letter 6,5                              ; load 'e'
        letter 6,14                             ; load 'n'

        RJMP    MoveOffTab




FoundBlue:
        SBI             PortB,4                 ; Turn on Blue LED

        RCALL   ClearLCD
        letter 4,2                              ; load 'B'
        letter 6,12                             ; load 'l'
        letter 7,5                              ; load 'u'
        letter 6,5                              ; load 'e'

        RJMP    MoveOffTab

MoveOffTab:
        LDI     Temp, 0x04              ; Moves robot forward for 1sec to get off tab
        LDI Temp2,0x89
        LDI Temp3, 0x7F
        OUT RSERVO_H,Temp
        OUT LSERVO_H,Temp
        OUT RSERVO_L,Temp2
        OUT LSERVO_L,Temp3


        LDI Temp,0b11110000
                                                ; bits7,6   =1,1 Set up and clear down (OC1A)
                                                ;    5,4    =1,1 Set up and clear down (OC1B)
                                                ; bits3,2   =0,0 Force output compare OFF A and B
                                                ; bits1,0   =0,0 PWM phase and Freq correct
        OUT TCCR1A,Temp

        RCALL   DelayL
        JMP             StartAgain


;**********************
;
;*   Knock over can   *
;**********************
;
SwingArm:
        ; Sets OCR0
        ;219=swing out
        ;245=home position

        LDI Temp,219
        OUT ARM_SERVO,Temp

        RET
```

```
;****************************************
;*   Line Track  With Obstacle Avoidance   *
;****************************************
;
;=============================================
; Full Clockwise        (Right Side)            : 1172 0x491
; Neutral                                                                       : 1156 0x484
; Full C. Clockwise     (Left Side)                     : 1141 0x477
;
; High if black or nothing
; Low if White
;=============================================
LineTrack:
            CLR                 ADCCounter                              ; Clear the ADC "object is too close" register
ADC_Obstacle_Check:
            IN                  Temp,ADCH                               ; Check A/D value
            CPI                 Temp,ADC_Distance
            BRGE        LineTrack2
            INC                 ADCCounter                              ; If object is too close, increment the counter

            CPI                 ADCCounter, 3
            BRNE        ADC_Obstacle_Check

            RCALL       StopForObstacle
            JMP                 LineTrack

LineTrack2:
;Check Right Sensor
            RCALL       Delay5ms
            RCALL       Delay5ms
            RCALL       Delay5ms

            RCALL       CheckForTabs
            SBIC        PINA,7
            RJMP        RightSeesBlack
            ;Right Sensor sees White
            ;Check Left Sensor
                        SBIC        PINA,6
                        RJMP        TurnLeft
                                    ;Left sees White
                                    RJMP        OffTrack

RightSeesBlack:
            ;Check Left Sensor
            SBIC        PINA,6
            RJMP        Straight
                        ;Left sees White
                        RJMP        TurnRight

TurnLeft:
            ; Sets Pulse On Period
            LDI         Temp, 0x04
            LDI Temp2,0x91
            LDI Temp3, 0x7F
```

```
        OUT RSERVO_H,Temp
        OUT LSERVO_H,Temp
        OUT RSERVO_L,Temp2
        OUT LSERVO_L,Temp3
        CLR prevIR
        RJMP LineTrack

TurnRight:
        ; Sets Pulse On Period
        LDI     Temp, 0x04
        LDI Temp2,0x89
        LDI Temp3, 0x7D
        OUT RSERVO_H,Temp
        OUT LSERVO_H,Temp
        OUT RSERVO_L,Temp2
        OUT LSERVO_L,Temp3
        CLR prevIR
        RJMP LineTrack
Straight:
        ; Sets Pulse On Period
        LDI     Temp, 0x04
        LDI Temp2,0x89
        LDI Temp3, 0x7F
        OUT RSERVO_H,Temp
        OUT LSERVO_H,Temp
        OUT RSERVO_L,Temp2
        OUT LSERVO_L,Temp3
        CLR prevIR
        RJMP LineTrack

Offtrack:
        INC             prevIR
        CPI             prevIR,210
        BRNE    LineTrack

        RCALL   ShutDown


;**********************
;*   Check for Tabs   *
;**********************
CheckForTabs:
        SBIC    PINA,5
        RJMP    TabDetected
        RET
TabDetected:
;       RCALL   ClearLCD
;       letter 5,4                              ; load 'T'
;       letter 6,1                              ; load 'a'
;       letter 6,2                              ; load 'b'

        RCALL   GetMean

FoundTab:
        RJMP FoundTab


;******************
;*   Shut Down   *
;******************
;Determines Which message to print on LCD screen

ShutDown:
        LDI     Temp,0b00000000     ; Disconnect PWM
        OUT     TCCR1A,Temp

        RCALL   ClearLCD
CheckIfOffTrack:
        CPI             prevIR,150
        BRNE    CheckIfBump
        RCALL   PrintOffTrack
        RJMP    RobotSleep
```

```
CheckIfBump:
        RCALL   PrintBump

RobotSleep:
        RJMP    RobotSleep




;**********************************
;*  Waits for obstacle to move   *
;**********************************
StopForObstacle:
        RCALL   ClearLCD
        RCALL   PrintObstacle

        RCALL   ClearTCNT
;Reverse 1.75sec
        LDI             Temp, 0x04
        LDI     Temp2,0x7F
        LDI     Temp3, 0x89
        OUT     RSERVO_H,Temp
        OUT     LSERVO_H,Temp
        OUT     RSERVO_L,Temp2
        OUT     LSERVO_L,Temp3

        RCALL   DelayL
        RCALL   DelayS
        RCALL   DelayS
        RCALL   DelayS

        RCALL   ClearTCNT
;Left 1 sec
        LDI             Temp, 0x04
        LDI             Temp2,0x91
        LDI     Temp3, 0x84
        OUT     RSERVO_H,Temp
        OUT     LSERVO_H,Temp
        OUT             RSERVO_L,Temp2
        OUT             LSERVO_L,Temp3

        RCALL   DelayL

        RCALL   ClearTCNT
;Straight 1 sec
        LDI     Temp, 0x04
        LDI Temp2,0x89
        LDI Temp3, 0x7F
        OUT RSERVO_H,Temp
        OUT LSERVO_H,Temp
        OUT RSERVO_L,Temp2
        OUT LSERVO_L,Temp3

        RCALL   DelayL

        LDI     Temp,0b00000000         ; Disconnect PWM
        OUT     TCCR1A,Temp

        RCALL   KnockOverObstacle

        RCALL   PWMInit_Wheels

;Reverse 1sec
        LDI     Temp, 0x04
        LDI Temp2,0x7F
```

```
            LDI Temp3, 0x89
            OUT RSERVO_H,Temp
            OUT LSERVO_H,Temp
            OUT RSERVO_L,Temp2
            OUT LSERVO_L,Temp3

            RCALL    DelayL

            RCALL    ClearTCNT



;Backwards Right 1.25sec
            LDI             Temp, 0x04
            LDI             Temp2,0x77
            LDI       Temp3, 0x84
            OUT       RSERVO_H,Temp
            OUT       LSERVO_H,Temp
            OUT             RSERVO_L,Temp2
            OUT             LSERVO_L,Temp3

            RCALL    DelayL
;           RCALL    DelayS

;Return to going straight
            LDI       Temp, 0x04
            LDI Temp2,0x89
            LDI Temp3, 0x7F
            OUT RSERVO_H,Temp
            OUT LSERVO_H,Temp
            OUT RSERVO_L,Temp2
            OUT LSERVO_L,Temp3

            RCALL    ClearLCD
            RET

;***********************************************
;*   Swings Out Arm To Knock Over An Obstacle   *
;***********************************************
KnockOverObstacle:

            LDI Temp, 0
            OUT TCNT0, Temp

            ; Sets OCR0
            ;219=swing out
            ;245=home position

            LDI Temp,219
            OUT ARM_SERVO,Temp

            RCALL    DelayL

            ; Start TCNT's at $00
            LDI Temp, 0
            OUT TCNT0, Temp

            ; Sets OCR0
            ;219=swing out
            ;245=home position

            LDI Temp,245
            OUT ARM_SERVO,Temp

            RET


;***********************
;*   Print Off Track   *
;***********************
```

```
PrintOffTrack:
        letter 4,15                             ; load 'O'
        letter 6,6                              ; load 'f'
        letter 6,6                              ; load 'f'
        letter 10,0                             ; load ' '
        letter 5,4                              ; load 'T'
        letter 7,2                              ; load 'r'
        letter 6,1                              ; load 'a'
        letter 6,3                              ; load 'c'
        letter 6,11                             ; load 'k'
        RET

;*******************
;*   Print Bump    *
;*******************
PrintBump:
        letter  4,2                             ; load "B"
        letter  7,5                             ; load "u"
        letter  6,13                            ; load "m"
        letter  7,0                             ; load "p"
        RET


;***********************
;*   Print Obstacle    *
;***********************
PrintObstacle:
        letter  4,15                            ; Load 'O'
        letter  6,2                                     ; Load 'b'
        letter  7,3                             ; Load 's'
        letter  7,4                             ; Load 't'
        letter  6,1                             ; Load 'a'
        letter  6,3                             ; Load 'c'
        letter  6,12                            ; Load 'l'
        letter  6,5                             ; Load 'e'
        RET


;*****************
;*   Clear LCD   *
;*****************
ClearLCD:
        LDI             LCDReg,0
        OUT     LCD_PORT,LCDReg
        CBI     LCD_PORT,5
        RCALL   Latch
        RCALL   Delay5ms
        LDI     LCDReg,1
        OUT     LCD_PORT,LCDReg
        CBI     LCD_PORT,5
        RCALL   Latch
        RCALL   Delay5ms
        RET


;*************
;*   Latch   *
;*************
;Used to Make A Falling Edge
Latch:
        SBI             LCD_PORT,4                      ; set E=1
        CBI     LCD_PORT,4                      ; set E=0
        RET


;************
;*   Send   *
;************
Send:

        SBIS    UCSRA,UDRE
        RJMP    Send
```

```
        OUT             UDR,Temp

        RET




;****************
;*  Receive1   *
;****************
Receive1:

        SBIS    UCSRA,RXC
        RJMP    Receive1
        IN              Temp,UDR
        STS             S1,Temp
        RET


;****************
;*  Receive2   *
;****************
Receive2:

        SBIS    UCSRA,RXC
        RJMP    Receive2
        IN              Temp,UDR
        STS             S2,Temp
        RET


;***************
;*  Receive3  *
;***************
Receive3:

        SBIS    UCSRA,RXC
        RJMP    Receive3
        IN              Temp,UDR
        STS             R_VALUE,Temp
        RET


;****************
;*  Receive4   *
;****************
Receive4:

        SBIS    UCSRA,RXC
        RJMP    Receive4
        IN              Temp,UDR
        STS             G_VALUE,Temp
        RET


;****************
;*  Receive5   *
;****************
Receive5:

        SBIS    UCSRA,RXC
        RJMP    Receive5
        IN              Temp,UDR
        STS             B_VALUE,Temp
        RET


;****************
;*  Receive6   *
```

```
;****************
Receive6:

        SBIS    UCSRA,RXC
        RJMP    Receive6
        IN              Temp,UDR
        STS             Rdev,Temp
        RET
;****************
;*  Receive7  *
;****************
Receive7:

        SBIS    UCSRA,RXC
        RJMP    Receive7
        IN              Temp,UDR
        STS             Gdev,Temp
        RET


;****************
;*  Receive8  *
;****************
Receive8:

        SBIS    UCSRA,RXC
        RJMP    Receive8
        IN              Temp,UDR
        STS             Bdev,Temp
        RET


;****************
;*  Receive9  *
;****************
Receive9:

        SBIS    UCSRA,RXC
        RJMP    Receive9
        IN              Temp,UDR
        STS             S9,Temp
        RET


;*************************
;*  Clears Wheel TCNT  *
;*************************
ClearTCNT:
        LDI Temp, 0
        OUT TCNT1H, Temp
        OUT TCNT1L, Temp
        RET
```

```
;****************
;*   Delay 1s   *
;****************
DelayL:
; ==============================
;    8000000 cycles:
; ------------------------------
; delaying 7999992 cycles:
            LDI         Delay1, $48
WGLOOP0:
            LDI                     Delay2, $BC
WGLOOP1:
            LDI                     Delay3, $C4
WGLOOP2:
            DEC                     Delay3
            BRNE        WGLOOP2
            DEC         Delay2
            BRNE        WGLOOP1
            DEC         Delay1
            BRNE        WGLOOP0
; ------------------------------
; delaying 6 cycles:
            LDI         Delay1, $02
WGLOOP3:
            DEC         Delay1
            BRNE        WGLOOP3
; ------------------------------
; delaying 2 cycles:
            NOP
            NOP
; ==============================
            RET


;********************
;*   Delay 0.25s   *
;********************
DelayS:
; ==============================
;    1000000 cycles:
; ------------------------------
; delaying 999999 cycles:
            LDI         Delay1,$09
WGLOOP00:
            LDI         Delay2,$BC
WGLOOP11:
            LDI         Delay3,$C4
WGLOOP22:
            DEC         Delay3
            BRNE        WGLOOP22
            DEC         Delay2
            BRNE        WGLOOP11
            DEC         Delay1
            BRNE        WGLOOP00
; ------------------------------
; delaying 1 cycle:
            NOP
; ==============================
RET
```

```
;****************
;*   Delay 5ms  *
;****************
Delay5ms:
; =============================
;    delay loop generator
;     40000 cycles:
; -----------------------------
; delaying 39999 cycles:
                        LDI  Delay1,$43
LoopD:
                        LDI  Delay2,$C6
LoopE:
                        DEC  Delay2
                        BRNE LoopE
                        DEC  Delay1
                        BRNE LoopD
; -----------------------------
; delaying 1 cycle:
                        NOP
; =============================
                        RET


;******************
;*   Delay 1.5ms  *
;******************
Delay1p5ms:
; =============================
;    delay loop generator
;     12000 cycles:
; -----------------------------
; delaying 11997 cycles:
                        LDI  Delay1, $1F
Delay1p5msLOOP0:
                        LDI  Delay2, $80
Delay1p5msLOOP1:
                        DEC  Delay2
                        BRNE Delay1p5msLOOP1
                        DEC  Delay1
                        BRNE Delay1p5msLOOP0
; -----------------------------
; delaying 3 cycles:
                        LDI  Delay1, $01
Delay1p5msLOOP2:
                        DEC  Delay1
                        BRNE Delay1p5msLOOP2
; =============================
                        RET
```