Milwaukee School of Engineering
CS391 – Embedded Computer System Design

Final Project Report

Student: **Marek Handl**

Date: May 20, 2007

Project objectives:

Create a small robot with wheels that would either follow a black line drawn on the floor or follow a light source. It shall be possible to choose from the two modes using buttons. Mode of operation shall be displayed on LCD.

Hardware equipment:

- Handy Board kit V1.21 (with MC68HC11E microcontroller)
- LCD - MTC16205D
- Line Tracking Sensor – Tracker Ver 2.0 from Lynxmotion, Inc.
- two Servo Motors – SO5/STD and wheels
- two cadmium sulphide photocells – CTST05
- computer with a serial port

Software equipment:

- JBug11 ver. 4.5.1.0524
- GNU 68HC1x compiler ver. 3.0 – using MSOE package from Dr. Durant
- any kind of text editor for writing the code
- everything developed and tested on Intel platform with MS Windows XP SP2

HW Components description:

Line Tracking Sensor

The sensor is composed of three pairs of infrared LED's and infrared sensors. The LED's are illuminated regularly and if there is a shiny surface underneath, the reflected IR light will be detected by the sensors. If there is a line (which is black), no light will get back to the sensor.

Outputs of the sensors go high when on white surface and go down when on black surface. The sensor needs to be powered by 5V DC.

Connecting to the board – connect the power cable to 0 and +5V, connect the data cable to digital ports 13, 14 and 15.

CdS Photocells

Cadmium sulphide photocell changes its resistance when exposed to visible light. It is possible to connect a photocell directly to handy board's analog inputs without any additional power source. When reading from the port, value between 0 and 255 is returned; higher value means less light.

Connecting to the board – connect the left sensor to analog port 2 and the right sensor to analog port 3.

LCD

Providing the LCD is connected to the board using standard pins, it is necessary to switch to single chip mode before each LCD read/write operation. To be able to operate with LCD in the single chip mode, the controlling routine has to be stored in zero page memory (extended RAM is not accessible in single chip mode). Once the character has been written to the LCD, it is possible to switch back to extended mode.

Motors

Only on/off state of motors is controlled in this program. No pulse width modulation has been used.

Connecting to the board – connect the left motor to motor 3 pins and the right motor to motor 2 pins.

Program description:

After necessary initialization the program gets into a never ending loop, which checks for a button press in each iteration. If no button was pushed, the board stays in the same state (either does nothing or continues in previous action).

When in menu (no-action) state, it is possible to choose from line tracking or follow light mode using the STOP button. To start the action, the START button needs to be pressed.

When in action state (either line tracking or following light), the STOP button can be used to stop the robot and return to the menu.

Line Tracking – all three sensors are read (left, center and right) and motors are switched on/off accordingly (left sensor sees black, left motor is switched off and right motor is switched on etc.)

Following Light – values from both sensors are read and compared. If the left sensor receives more light (value is lower than from the right sensor), left motor is switched off and right motor is switched on and vice versa.

Design decisions:

Visible light sensors are connected to analog ports

It is necessary to compare the two values (from the right and from the left sensor). The value on input is converted to 0-255 range using A/D conversion. A two-state value (a zero or a one), which would be returned from a digital port, would not be enough.

Line tracking sensors are connected to digital ports

The sensors can return only 2 values (either IR light was reflected or was not). There is no point in connecting these sensors to analog ports. It is possible to configure the Handy Board to use the analog ports as digital inputs, but this way it is easier.

LCD operating routine in zero page memory

Necessary because of the single chip mode. It is not possible to operate the LCD in extended chip mode, unless additional chips and wiring are used.

Using never ending checking for button press

It would be possible to use interrupts, but presented solution is simpler and equally effective.

No pulse width modulation for motors

All required tasks can be performed using on/off state only. There is no need for using pulse width modulation.

Source code:

```
* *******************************************************
* REGISTERS *********************************************
* *******************************************************
; addresses for single chip mode - index addressing will be used
sPORTA        = 0x00
sPORTB        = 0x04
sPORTC        = 0x03
sHPRIO        = 0x3C
sDDRC         = 0x07
sSPCR         = 0x28

; addresses for extended mode - direct addressing will be used
PORTA         = 0x1000
PORTB         = 0x1004
PORTC         = 0x1003
HPRIO         = 0x103C      ; Highest Priority Interrupt and misc.
DDRC          = 0x1007      ; Data Direction register for port C

SPCR          = 0x1028      ; SPI control Register
SPSR          = 0x1029
SPDR          = 0x102A      ; SCI Data Register
BAUD          = 0x102B      ; SCI Baud Rate Control Register
SCCR1         = 0x102C      ; SCI Control Register 1
SCCR2         = 0x102D      ; SCI Control Register 2
SCSR          = 0x102E      ; SCI Status Register
SCDR          = 0x102F      ; SCI Data Register

MOTORS        = 0x7000  ; motors output
DIGIN         = 0x7FFF  ; Digital input

* *******************************************************
* CONSTANTS *********************************************
* *******************************************************
lefton        = 0x80    ; left motor on
righton       = 0x40    ; right motor on
bothon        = 0xC0    ; left and right motor on

startbt       = 0b10000000  ; start button, active in low
stopbt        = 0b01000000  ; stop button, active in low

option        = 0x1039     ; A/D setup
ADPU          = (1 << 7)   ; A/D setup
adctl         = 0x1030      ; A/D control register
adr1          = 0x1031       ; result from analog port 0
adr2          = 0x1032       ; result from analog port 1
adr3          = 0x1033       ; result from analog port 2
adr4          = 0x1034       ; result from analog port 3

single        = 0b00100000 ; single chip mode
```

```
TDRE            = 0x80          ; Transmit Data Register Empty
TRENA           = 0x0C          ; Transmit, Receive ENAble
RDRF            = 0x20          ; Receive Data Register Full
PD_WOM          = 0x20
brate           = 0xB0          ; Baud Rate

lcd_routine     = 0x10 ; adress of lcd routine (it's in zero page memory because of single chip mode)
lcd_temp        = 0x09 ; address of a temp variable in single chip mode

state_lt        = 1    ; line tracker
state_fl        = 2    ; follow light

* ********************************************************
* DATA VARIABLES *****************************************
* ********************************************************
.section .bss
state:          .rmb 1          ; 0-menu, 1-line tracker, 2-follow light
menu_item:      .rmb 1    ; 0-line tracker, 1-follow light
motors_state:   .rmb 1
pointer:        .rmb 2    ; pointer to a charecter being displayed
count:          .rmb 1    ; how many characters have been displayed so far
temp:           .rmb 1    ; used for counting

* ********************************************************
* PREDEFINED STRINGS *************************************
* ********************************************************
.section .rodata        ; strings are null terminated
menu_header:    .asciz  "----- MENU -----"
menu_lt:        .asciz  "1) Line Tracker"
menu_fl:        .asciz  "2) Follow Light"


* ********************************************************
* PROGRAM ************************************************
* ********************************************************
.section .text
.global _start

_start:
        ; initialization
        lds       #_stack
        clr     MOTORS
        clr     state
        clr     motors_state
        clr       count
        clr       menu_item

        ; A/D initialization
        ldaa    option
        oraa    #ADPU       ; power up A/D system
        staa    option
        ldaa    #0b00110000 ; scan of four ports of PE4
        staa    adctl

        ; lcd initialization
        ldx       #0x1000
        bclr     sSPCR, X PD_WOM

        ldaa     #brate
        staa     BAUD       ; set up baud rate
        ldaa     #TRENA
        staa     SCCR2

        jsr       copy_routine        ; copy LCD writing routine to zero page

        clra                        ; command
        ldab     #0x0C                      ; Display On /  Cursor Off / Flash Off
        jsr       lcd_routine

        clra                        ; command
        ldab     #0x38                      ; two line display
        jsr       lcd_routine

        jsr       clear
        ldaa     #1
        staa     menu_item
        jsr       change_menu

        ; ****************************************************
        ; loop checking for a pressed button
loop:
        ldaa    DIGIN    ; read digital input
        psha
        anda     #stopbt                ; if equals zero, stop button was pushed
        beq       stop_bt
        pula
        anda     #startbt       ; if equals zero, start button was pushed
        beq       start_bt

        ; no button pushed, keep the state
        ldaa     state
        beq       loop            ; state=0 means menu - wait for button press
        cmpa    #state_lt
        beq       work_lt                ; do line tracker
        cmpa    #state_fl
        beq       work_fl                ; do follow light
        bra       loop              ; unreachable code, just in case

stop_bt:
        ; stop button pushed
        ldaa     DIGIN
        anda     #stopbt
        beq       stop_bt                ; wait for button release
        ldaa    state
        bne      1f                  ; state=0 means menu
        jsr      change_menu
        bra      loop
1:
        ; stop action
        clr      MOTORS              ; stop motors
        clr      state           ; menu state
        jsr      change_menu
        bra      loop

start_bt:
        ; start button pushed
        ldaa    DIGIN
        anda    #startbt
        beq      start_bt       ; wait for button release

        ldaa    state
        beq      1f                 ; state=0 means menu - start working
        bra      loop            ; pushing start button while working doesn't change a thing

1:
  ; start action (either line tracker or follow light)
        jsr      clear           ; clear display
        ldaa    #bothon
        staa    MOTORS              ; switch both motors on

        ldaa    menu_item
        inca
        staa    state
        deca
        beq      start_lt       ; menu_item=0 means line tracker

  ; start follow light
        ldx      #menu_fl
        stx      pointer
        jsr      print
        jsr      linetracker
        bra      loop
start_lt:
        ; start line tracker
  ldx       #menu_lt
        stx      pointer
        jsr      print
        jsr      followlight
        bra     loop

work_lt:
  ; do line tracker
        jsr      linetracker
        bra      loop
work_fl:
  ; do follow light
        jsr        followlight
        bra        loop


* **************************************
* CHANGE MENU
* display "Menu" in upper row and menu item in lower row of LCD
* called after STOP button was pushed
* **************************************
change_menu:
        jsr        clear
        ldx        #menu_header
        stx        pointer
        jsr        print
        ;jsr       new_line      ; new line is automatically appended, because the first line is full
        ldaa      menu_item
        beq        1f
        dec        menu_item  ; change menu_item to zero (line tracker)
        ldx        #menu_lt
        stx        pointer
        jsr        print
        rts
1:
  inc        menu_item  ; change menu_item to one (follow light)
        ldx        #menu_fl
        stx        pointer
        jsr        print
        rts


* **************************************
* LINE TRACKER
* follows a black line drawn on the floor
* reads from Line Tracker Sensor and controls motors accordingly
* **************************************
linetracker:

  ldd   DIGIN        ; load to A from digital input
  anda  #0b00111000   ; bit5 = Left, bit4 = Center, bit3 = Right

  cmpa  #0b00111000    ; all white
  bne   left

  ; all white
  ldab  motors_state     ; previous state of motors
  cmpb  #0b00101000      ; both motors on
  bne   1f
  ldab  #lefton          ; switch off the right motor
  bra   done
1:
  ldab  motors_state    ; both motors off
  bne   2f
  ldab  #bothon         ; switch on both motors
2:
  ; one motor was on, keep moving in the same direction
  bra   done

left:
  ; check left sensor
  ldab  motors_state
  anda  #0b00101000    ; ignore center sensor
  cmpa  #0             ; left and right both black ... keep moving in the same direction
  beq   done

  clrb                 ; clear B - temporary place for motor directions
  psha
  anda  #0b00100000    ; left sensor
  beq   right
  orab  #lefton         ; turn on left motor

right:
  ; check right sensor
  pula
  anda  #0b00001000    ; right white
  beq   done
  orab  #righton         ; turn on right motor

done:
  ; set up motors
  stab  MOTORS
  stab  motors_state    ; remember motors directions
  rts



* **************************************************
* FOLLOW LIGHT
* follows a light source
* reads from two photocells, which are mounted on sides of the bot
* goes to that direction where more light comes from
* *************************************************
followlight:
  ldaa  adr3    ; left sensor
  cmpa  adr4    ; compare it to right sensor value
  blo   goright  ; branch if lower
  bgt   goleft  ; branch if greater

  ; same value - go straight
  ldaa  #bothon
  bra   9f

goleft:
  ; going to the left
  ldaa  #righton
  bra   9f

goright:
  ; going to the right
  ldaa  #lefton

9:
  staa  MOTORS
  jsr   wait
  rts

wait:
  ldx   #0xFFFF
1:
  dex
  bne   1b
  rts


* **************************************************
* LCD FUNCTIONS
* **************************************************

* ******************************************************
* PRINT A STRING TO LCD
* Starting address of the string in "pointer"
* String is terminated with a null character
* ******************************************************
print:
        ldx      pointer

        ldaa     #0x02           ; print command
        ldab     0, X            ; character to be displayed
        beq        9f            ; string is terminated with a "00" character

        inx
        stx        pointer
        jsr        lcd_routine   ; print the character to the lcd display

        inc        count
        ldaa     count
        cmpa   #16      ; first line is full
        bne        8f
        jsr        nl_fill

8:      bra      print
9:      rts

* ******************************************************
* NEW LINE Character - move to the second line
* uses "temp" as a temp variable
* ******************************************************
new_line:
        ldaa    #16
        suba    count         ; add spaces to the end of the line
        adda    #24
        staa    temp
        bra       1f
nl_fill:
        ldaa    #24     ; number of character between the end of the 1st line and the beginning of the 2nd line
        staa    temp
1:
  ldaa    #0x02         ; print command
        ldab    #32           ; space character
        jsr       lcd_routine
        dec       temp
        bne       1b
        clr       count
        rts


* ******************************************************
* CLEAR DISPLAY
* Moves cursor to the beginning too
* ******************************************************
clear:
        clra
        ldab    #0x01         ; Clear Display
        jsr       lcd_routine
        clra
        ldab    #0x02
        jsr       lcd_routine   ; Move cursor home
        clr     count
        rts

* ******************************************************
* RESET CURSOR
* Moves cursor to the beginning of the 1st line
* ******************************************************
home:
        clra
        ldab    #0x02         ; move cursor to beginning of 1st line
        jsr       lcd_routine
        clr       count
        rts

* ******************************************************
* COPY LCD ROUTINE to zero page memory
* necessary for single chip mode
* ******************************************************
copy_routine:
        ldx       #lcd_print     ; address in extended RAM
        ldy       #lcd_routine   ; target address in zero page memory
```

```
copy_loop:
        ldaa    0,x
        staa    0,y
        inx
        iny
        cpx       #routine_end
        bne       copy_loop
        rts

* *****************************************************
* LCD ROUTINE - prints out a character
* command is in acc A
* character is in acc B
* *****************************************************
lcd_print:
        sei
        staa    lcd_temp        ; save command value

        ldx       #0x1000
        bclr    sHPRIO, X 0b00100000        ; switch to single chip mode
        bclr    sPORTA, X 0b00010000        ; turn off LCD E line

        clr       sDDRC, X    ; port C as input

lcd_busy:
        ldaa    #1
        staa    sPORTB, X               ; read operation from LCD

        bset    sPORTA, X 0b00010000        ; frob LCD on
        ldaa    sPORTC, X               ; get status
        bclr    sPORTA, X 0b00010000        ; frob LCD off

        anda    #0x80                   ; busy flag
        bne       lcd_busy              ; wait for LCD ready

        ldaa    #0xFF
        staa    sDDRC, X                ; port C as output
        ldaa    lcd_temp
        staa    sPORTB, X               ; command (only the 2 LSB are important - R/W and RS bits)
        stab    sPORTC, X               ; data (bits DB0-DB7)

        bset    sPORTA, X 0b00010000
        bclr    sPORTA, X 0b00010000        ; frob LCD
        bset    sHPRIO, X 0b00100000        ; switch back to extended mode

        cli
        rts

routine_end: nop        ; used just as a pointer to the end of the routine
```