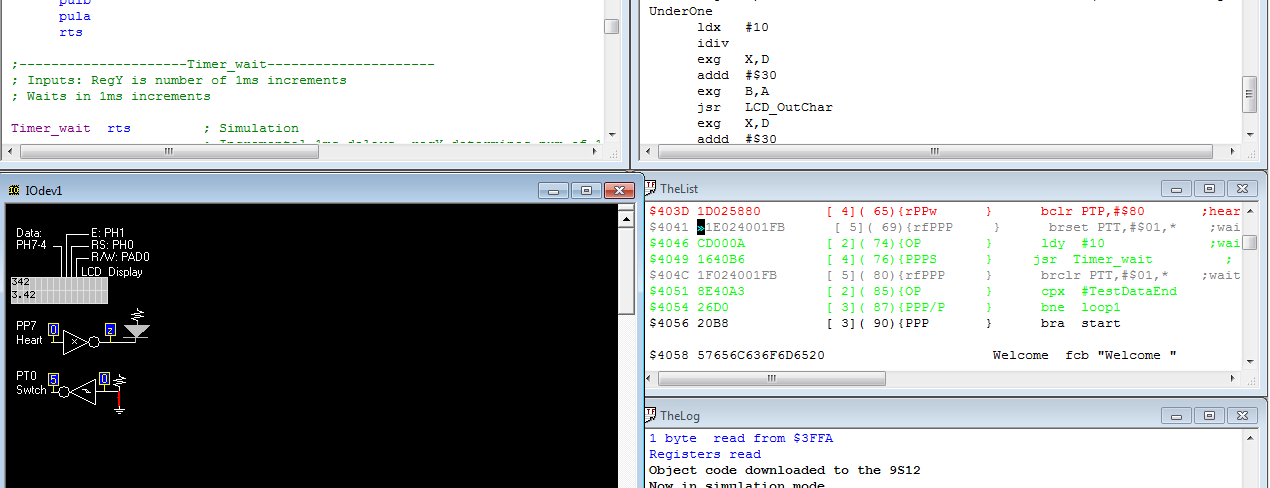
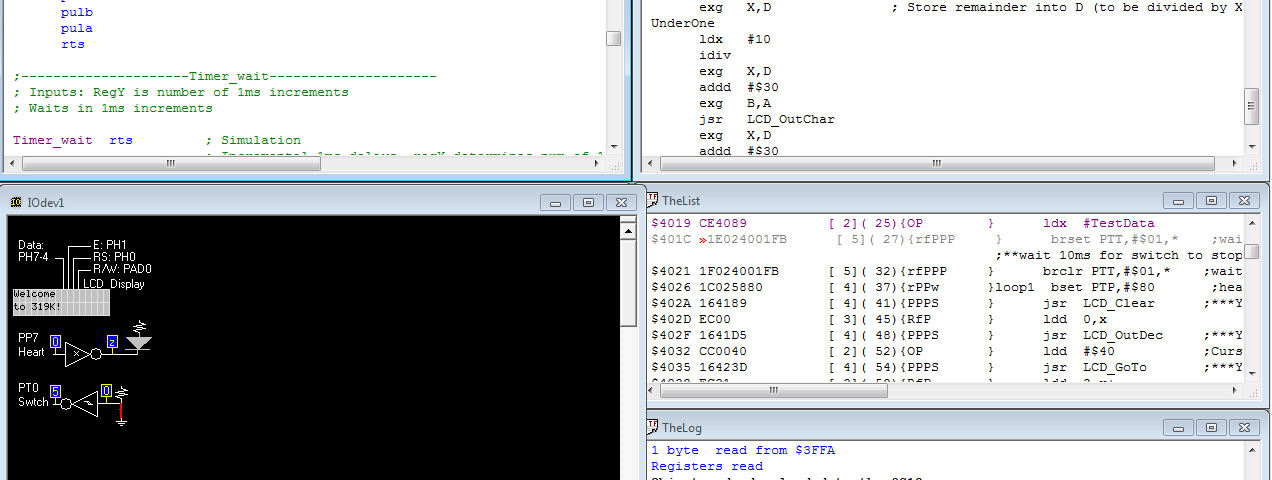
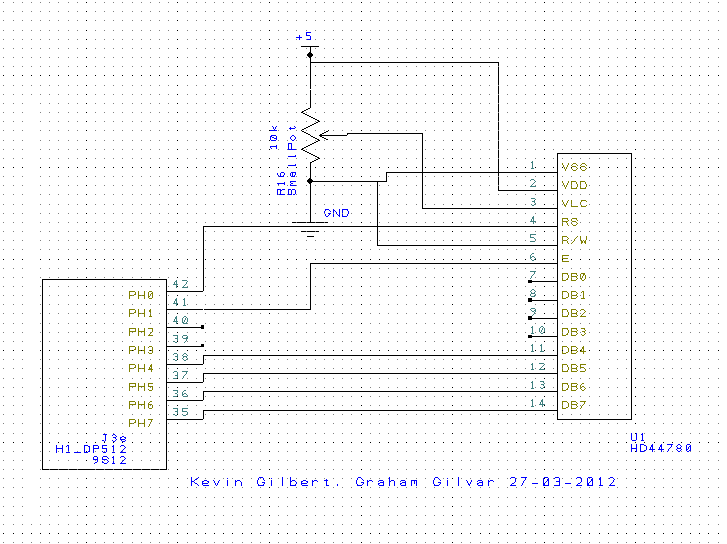
Kevin Gilbert and Graham Gilvar

27 March 2012

EE319K Lab

Lab 6





; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Lab6\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Kevin Gilbert and Graham Gilvar

; Date Created 23-03-2012 10:30 PM

; Date last modified: 29-03-2012 3:15PM

; LCD Device Driver

; I/O Directories

PTH equ $0260

DDRH equ $0262

PTP equ $0258

DDRP equ $025A

PTT equ $0240

DDRT equ $0242

TSCR1 equ $0046

TSCR2 equ $004D

TCNT equ $0044

; RAM global variables

org $2000

TCNTo rmb 2

StringPnt rmb 2

number rmb 2

point rmb 2

count rmb 2

counter rmb 2

numberoutfix rmb 2

compareoutfix rmb 2

baseoutfix rmb 2

printoutfix rmb 2

DecPnt rmb 2

DecCounter rmb 2

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Main \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

org $4000

main lds #$4000

bclr DDRT,#$01 ;PT0 digital input

bset DDRP,#$80 ;PP7 digital output, heartbeat to LED Page 32 Introduction to Embedded Systems: Interfacing to the Freescale 9S12

jsr LCD\_Open ;\*\*\*Your function that initializes the LCD\*\*\*

cli

start jsr LCD\_Clear ;\*\*\*Your function that clears the display\*\*\*

ldd #Welcome

jsr LCD\_OutString ;\*\*\*Your function that outputs a string\*\*

ldx #TestData

brset PTT,#$01,\* ;wait for switch release

;\*\*wait 10ms for switch to stop bouncing\*\*

brclr PTT,#$01,\* ;wait for switch touch

loop1 bset PTP,#$80 ;heartbeat

jsr LCD\_Clear ;\*\*\*Your function that clears the display\*\*\*

ldd 0,x

jsr LCD\_OutDec ;\*\*\*Your function that outputs an integer\*\*\*

ldd #$40 ;Cursor location of the 8th position

jsr LCD\_GoTo ;\*\*\*Your function that moves the cursor\*\*\*

ldd 2,x+

jsr LCD\_OutFix ;\*\*\*Your function that outputs a fixed-point\*\*\*

bclr PTP,#$80 ;heartbeat

brset PTT,#$01,\* ;wait for switch release

ldy #10 ;wait 10ms for switch to stop bouncing

jsr Timer\_wait ; Was bsr, changed to jsr for testing

brclr PTT,#$01,\* ;wait for switch touch

cpx #TestDataEnd

bne loop1

bra start

Welcome fcb "Welcome "

fcb " " ; 32 spaces

fcb "to 319K!",0

TestData fdb 0,1,9,73,99,162,342,543,777,999,1000,34567,65535

TestDataEnd

;---------------------Timer\_Init---------------------

; Inputs: NA

; Outputs: NA

; Initialize timer routine, initialize hardware counter

Timer\_Init

psha ; Save Registers

pshb

pshx

pshy

movb #$80,TSCR1

movb #$07,TSCR2 ; Set Hardware timer, 00 for simulation (faster)

puly ; Restore Registers

pulx

pulb

pula

rts

;---------------------Timer\_wait---------------------

; Inputs: RegY is number of 1ms increments

; Waits in 1ms increments

Timer\_wait ;rts ; Simulation

; Incremental 1ms delays, regY determines num of 1ms delays

psha ; Save Registers

pshb

pshx

pshy

movw TCNT,TCNTo

wait1ms adda #0

compareT ldd TCNT

subd TCNTo

cpd #65

blo compareT

dey

bne wait1ms

puly ; Restore Registers

pulx

pulb

pula

rts

;---------------------LCD\_Open---------------------

; initialize the LCD display, called once at beginning

; Input: none

; Output: none

; Registers modified: CC

LCD\_Open

pshd ; Save Registers

pshx

pshy

movw #0,StringPnt ; Initialize global variables

movw #0,number

jsr Timer\_Init

bset DDRH,#$FF ; All pins output, PH0 = RS, PH1 = E

ldaa #$03

jsr OutCsrNibble

ldy #5 ; 5ms delay

jsr Timer\_wait

ldaa #$03

jsr OutCsrNibble

ldy #1

jsr Timer\_wait

ldaa #$03

jsr OutCsrNibble

ldy #1

jsr Timer\_wait

ldaa #02

jsr OutCsrNibble

ldy #1

jsr Timer\_wait

ldaa #$28

jsr OutCsr

ldaa #$14

jsr OutCsr

ldaa #$06

jsr OutCsr

ldaa #$0C

jsr OutCsr

puly ; Restore Registers

pulx

puld

rts

;--------------- outCsrNibble ------------------

; sends 4 bits to the LCD control/status

; Input: RegA is 4-bit command, in bit positions 3,2,1,0 of RegA

; Output: none

OutCsrNibble ; Proto

psha

bclr PTH,#$02 ; E bit 0

lsla

lsla

lsla

lsla

anda #$F0

adda #$02 ; Keep E bit toggled to 1

staa PTH

; 10 nanosecond delay for real board runs if required

bclr PTH,#$02 ; Toggle E bit to 0

pula

ldy #20

jsr Timer\_wait

rts

;---------------------outCsr---------------------

; sends one command code to the LCD control/status

; Input: RegA is 8-bit command to execute

; Output: non

OutCsr

psha ; psha twice, restored twice

psha

bclr PTH,#$01 ; Set RS bit to be low

anda #$F0

lsra

lsra

lsra

lsra ; Send high 4 bits from RegA into Nibble in low 4 bit position

jsr OutCsrNibble ; Sends high 4 bits of A into CsrNibble

pula

anda #$0F

jsr OutCsrNibble ; Send low 4 bits of A into CsrNibble

pula ; Restore RegA

rts

; ---------------------LCD\_OutString-------------

; Output character string to LCD display, terminated by a NULL(0)

; Inputs: RegD (call by reference) points to a string of ASCII characters

; Outputs: none

; Registers modified: CC

LCD\_OutString

pshd

std StringPnt ; need to move the pointer to register X and the character to

loop ldx StringPnt ; register D since the subroutine LCD\_OutChar receives the

ldd 0,x ; character from register D.

cpd #0 ; checks for the null character in the string

lbeq done2

jsr LCD\_OutChar

tfr B,A ; Load next ASCII char into regA

jsr LCD\_OutChar

inx

inx

stx StringPnt

;jsr LCD\_GoTo ; goes to LCD\_GoTo subroutine to move the cursor

bra loop ; branches until null character is reached

done2

puld

ldy #20

jsr Timer\_wait

rts

;---------------------LCD\_Clear---------------------

; clear the LCD display, send cursor to home

; Input: none

; Outputs: none

; Registers modified: CC

LCD\_Clear

ldaa #$01

bclr PTH,#$01 ; RS 0, sending command

jsr OutCsr ; clears display

ldy #2

jsr Timer\_wait

ldaa #$02

jsr OutCsr

ldy #2

jsr Timer\_wait

rts

;---------------------LCD\_OutChar---------------------

; sends one ASCII to the LCD display

; Input: RegD (call by value) letter is 8-bit ASCII code

; Outputs: none

; Registers modified: CC

LCD\_OutChar

pshd

pshd

bset PTH,#$01 ; Set RS bit to 1 to send data, E bit 0

anda #$F0

adda #$03 ; high 4 bits stored, RS = 1, E = 1

staa PTH

suba #$02 ; RS = 1, E = 0, toggle E bit

staa PTH

ldy #20

jsr Timer\_wait

puld ; Restore RegD

anda #$0F

lsla

lsla

lsla

lsla

adda #$03

staa PTH

suba #$02 ; Toggle E bit

staa PTH

puld

ldy #20

jsr Timer\_wait

rts

;-----------------------LCD\_OutDec-----------------------

; Output a 16-bit number in unsigned decimal format

; Input: RegD (call by value) 16-bit unsigned number

; Output: none

; Registers modified: CCR

LCD\_OutDec

pshd

pshx

ldx #10

idiv

cpx #0

beq Decdone

exg D,X

jsr LCD\_OutDec

exg X,D

Decdone

addd #$0030 ; #'0'

exg B,A

jsr LCD\_OutChar

pulx

puld

ldy #20

jsr Timer\_wait

rts

; -----------------------LCD\_OutFix----------------------

; Output characters to LCD display in fixed-point format

; unsigned decimal, resolution 0.01, range 0.00 to 9.99

; Inputs: RegD is an unsigned 16-bit number

; Outputs: none

; Registers modified: CCR

; E.g., RegD=0, then output "0.00 "

; RegD=3, then output "0.03 "

; RegD=89, then output "0.89 "

; RegD=123, then output "1.23 "

; RegD=999, then output "9.99 "

; RegD>999, then output "\*.\*\* "

LCD\_OutFix

cmpd #1000

blo good

pshd ; Display "\*.\*\*" if greater than 1000

;-- movb #0,1,-SP

;-- movb #0,1,-SP

;-- movb #'\*',1,-SP

;-- movb #'\*',1,-SP

;-- movb #'.',1,-SP

;-- movb #'\*',1,-SP

ldaa #'\*'

jsr LCD\_OutChar

ldaa #'.'

jsr LCD\_OutChar

ldaa #'\*'

jsr LCD\_OutChar

ldaa #'\*'

jsr LCD\_OutChar

;-- ldd 6,SP+

puld

rts

good

pshd

pshx

ldx #100

idiv

exg X,D

jsr LCD\_OutDec

ldd #$002E ; Period

exg B,A

jsr LCD\_OutChar

cpx #10

blo extrao

Fixcontinue

exg D,X

jsr LCD\_OutDec

pulx

puld

rts

extrao

ldaa #'0'

jsr LCD\_OutChar

bra Fixcontinue

;-----------------------LCD\_GoTo-----------------------

; Move cursor (set display address)

; Input: RegD is display address is 0 to 7, or $40 to $47

; Output: none

; errors: it will check for legal address

LCD\_GoTo

pshd ; Save Registers

pshx

pshy

cmpd #$07

bhi ErrChcklo ; If greater than #$07, Check if less than $3F

resume

cmpd #$49

bhi ErrChckhi

clra

tfr B,A ; Copy value into regA

adda #$80

jsr OutCsr

bra returnErr ; restore Reg and return

ErrChcklo

cmpd #$40

blo returnErr ; Between $08 and $3F, return error (nothing)

bra resume

ErrChckhi

cmpd #$FF

bls returnErr ; Between $49 and $FF

returnErr

puly ; Restore Registers

pulx

puld

ldy #20

jsr Timer\_wait

rts

org $FFFE ; Reset Vector

fdb main