;;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;; FILENAME: LCD.asm

;; Version: 1.5, Updated on 2010/6/8 at 12:37:12

;; Generated by PSoC Designer 5.4.3191

;;

;; DESCRIPTION: LCD User Module software implementation file.

;;

;; This set of functions is written for the common 2 and 4 line

;; LCDs that use the Hitachi HD44780A controller.

;;

;; LCD connections to PSoC port

;;

;; PX.0 ==> LCD D4

;; PX.1 ==> LCD D5

;; PX.2 ==> LCD D6

;; PX.3 ==> LCD D7

;; PX.4 ==> LCD E

;; PX.5 ==> LCD RS

;; PX.6 ==> LCD R/W

;;

;; NOTE: User Module APIs conform to the fastcall16 convention for marshalling

;; arguments and observe the associated "Registers are volatile" policy.

;; This means it is the caller's responsibility to preserve any values

;; in the X and A registers that are still needed after the API functions

;; returns. For Large Memory Model devices it is also the caller's

;; responsibility to preserve any value in the CUR\_PP, IDX\_PP, MVR\_PP and

;; MVW\_PP registers. Even though some of these registers may not be modified

;; now, there is no guarantee that will remain the case in future releases.

;;-----------------------------------------------------------------------------

;; Copyright (c) Cypress Semiconductor 2010. All Rights Reserved.

;;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

include "m8c.inc"

include "memory.inc"

include "LCD.inc"

;-----------------------------------------------

; Global Symbols

;-----------------------------------------------

export LCD\_Start

export \_LCD\_Start

export LCD\_Init

export \_LCD\_Init

export LCD\_WriteData

export \_LCD\_WriteData

export LCD\_Control

export \_LCD\_Control

export LCD\_PrString

export \_LCD\_PrString

export LCD\_PrCString

export \_LCD\_PrCString

export LCD\_Position

export \_LCD\_Position

export LCD\_PrHexByte

export \_LCD\_PrHexByte

export LCD\_PrHexInt

export \_LCD\_PrHexInt

export LCD\_Delay50uTimes

export \_LCD\_Delay50uTimes

export LCD\_Delay50u

export \_LCD\_Delay50u

;-----------------------------------------------

; If bargraph functions not required, don't

; export the function names.

;-----------------------------------------------

IF (LCD\_BARGRAPH\_ENABLE)

export LCD\_InitBG

export \_LCD\_InitBG

export LCD\_InitVBG

export \_LCD\_InitVBG

; NOTE: The two functions,

;

; LCD\_DrawVBG and

; LCD\_DrawBG

;

; are implemented using both fastcall16 and legacy fastcall16 because they

; fall into a special and rare case where the calling sequences specified

; by the two disciplines are incompatible. The fastcall16 versions are

; provided for both C and Assembly users in all memory models. The legacy

; fastcall16 versions are provided only to support existing small memory

; model assembly language code---they do not work in the large memory

; model.

;

; \*\* The legacy fastcall16 versions are provided on a temporary basis to

; \*\* ease the transition to the 4.2 release of PSoC Designer. Their use is

; \*\* deprecated and their status is "No Further Maintenance".

;

; The fastcall16 versions of these functions are distinguished by a

; leading underscore in the name. The legacy fastcall16 names (which appear

; in this comment) do not have the leading underscore. Details on the

; calling sequence to be used for fastcall16 are given in the user module

; datasheet.

;

; Fastcall16 versions:

export \_LCD\_DrawVBG

export \_LCD\_DrawBG

IF SYSTEM\_SMALL\_MEMORY\_MODEL

; Legacy Fastcall versions:

export LCD\_DrawVBG

export LCD\_DrawBG

ENDIF ; SYSTEM\_SMALL\_MEMORY\_MODEL

ENDIF ; BARGRAPH\_ENABLE

;

; The following functions are deprecated and will be eliminated in a future

; version of PSoC Designer.

;

export LCD\_Write\_Data

export \_LCD\_Write\_Data

;-----------------------------------------------

; EQUATES

;-----------------------------------------------

LCD\_Port: equ PRT2DR

LCD\_PortMode0: equ PRT2DM0

LCD\_PortMode1: equ PRT2DM1

LCD\_E: equ 10h

LCD\_RW: equ 40h

LCD\_RS: equ 20h

LCD\_DATA\_MASK: equ 0Fh

LCD\_READY\_BIT: equ 08h

LCD\_DATA\_READ: equ ( LCD\_E | LCD\_RW | LCD\_RS )

LCD\_CNTL\_READ: equ ( LCD\_E | LCD\_RW )

LCD\_PORT\_WRITE: equ 7Fh

LCD\_PORT\_MASK: equ 7Fh

LCD\_DISP\_INC: equ 03h

LCD\_DISP\_OFF: equ 08h

LCD\_DISP\_ON: equ 0Ch

LCD\_4BIT\_2LINE: equ 2Ch

;-----------------------------------------------

; Bargraph definitions

;-----------------------------------------------

LCD\_BG\_CHAR\_WIDTH: equ 16 ; 16 characters in width

LCD\_BG\_SEG\_WIDTH: equ 80 ; 16 \* 5 = 80

LCD\_BG\_COL\_START: equ 0 ; Always start in the left most column

; Offsets for 2x16, 2x20, 4x20

; Change these values for a custom LCD

LCD\_ROW1\_OFFSET: equ 80h ; Address/command offset for row 1

LCD\_ROW2\_OFFSET: equ C0h ; Address/command offset for row 2

LCD\_ROW3\_OFFSET: equ 94h ; Address/command offset for row 1

LCD\_ROW4\_OFFSET: equ D4h ; Address/command offset for row 2

LCD\_BG\_ROW1\_OFFSET: equ 80h ; Address/command offset for row 1

LCD\_BG\_ROW2\_OFFSET: equ C0h ; Address/command offset for row 2

LCD\_CG\_RAM\_OFFSET: equ 40h ; Offset to character RAM

AREA UserModules (ROM, REL)

.SECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_PrCString

;

; DESCRIPTION:

; Print constant (ROM) string to LCD

;-----------------------------------------------------------------------------

;

; ARGUMENTS:

; A:X Pointer to String

; A contains MSB of string address

; X contains LSB of string address

;

; RETURNS: none

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

; Currently only the page pointer registers listed below are modified:

; CUR\_PP

;

LCD\_PrCString:

\_LCD\_PrCString:

RAM\_PROLOGUE RAM\_USE\_CLASS\_1

.Loop\_PrCString:

push A ; Store ROM pointer

push X

romx ; Get character from ROM

jnz .LCD\_PrCString\_WR ; print character and advance pointer

pop X ; Restore the stack

pop A

RAM\_EPILOGUE RAM\_USE\_CLASS\_1

ret ; Return

.LCD\_PrCString\_WR:

call LCD\_WriteData ; Write data to LCD

pop X ; Get ROM pointer

pop A

inc X ; Inc LSB of pointer

jnc .Loop\_PrCString

inc A ; Inc MSB of pointer if LSB overflow

jmp .Loop\_PrCString

.ENDSECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_PrHexByte

;

; DESCRIPTION:

; Print a byte in Hex (two characters) to current LCD position

;

;-----------------------------------------------------------------------------

;

; ARGUMENTS:

; A => (BYTE) Data/char to be printed

;

; RETURNS: none

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

; Currently only the page pointer registers listed below are modified:

; CUR\_PP

;

.LITERAL

LCD\_HEX\_STR::

DS "0123456789ABCDEF"

.ENDLITERAL

.SECTION

LCD\_PrHexByte:

\_LCD\_PrHexByte:

RAM\_PROLOGUE RAM\_USE\_CLASS\_1

push A ; Save lower nibble

asr A ; Shift high nibble to right

asr A

asr A

asr A

and A,0Fh ; Mask off nibble

index LCD\_HEX\_STR ; Get Hex value

call LCD\_WriteData ; Write data to screen

pop A ; Restore value

and A,0Fh ; Mask off lower nibble

index LCD\_HEX\_STR ; Get Hex value

call LCD\_WriteData ; Write data to screen

RAM\_EPILOGUE RAM\_USE\_CLASS\_1

ret

.ENDSECTION

.SECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_PrHexInt

;

; DESCRIPTION:

; Print an Int in Hex (four characters) to current LCD position

;

;-----------------------------------------------------------------------------

;

; ARGUMENTS:

; A:X Integer value

; A contains LSB of Int

; X contains MSB of Int

;

; RETURNS: none

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

; Currently only the page pointer registers listed below are modified:

; CUR\_PP

;

LCD\_PrHexInt:

\_LCD\_PrHexInt:

RAM\_PROLOGUE RAM\_USE\_CLASS\_1

swap A,X

call LCD\_PrHexByte ; Print MSB

mov A,X ; Move LSB into position

call LCD\_PrHexByte ; Print LSB

RAM\_EPILOGUE RAM\_USE\_CLASS\_1

ret

.ENDSECTION

.SECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_PrString

;

; DESCRIPTION:

; Print (RAM) ASCII string to LCD

;

;-----------------------------------------------------------------------------

;

; ARGUMENTS:

; A:X contains pointer to string

; X contains LSB of string pointer

; A contains MSB or page of string pointer (not used at this time)

;

; RETURNS:

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

; Currently only the page pointer registers listed below are modified:

; CUR\_PP

; IDX\_PP

;

;

LCD\_PrString:

\_LCD\_PrString:

RAM\_PROLOGUE RAM\_USE\_CLASS\_3

RAM\_SETPAGE\_IDX A

.Loop\_PrString:

mov A,[X] ; Get value pointed to by X

jz .End\_LCD\_PrString ; Check for end of string

;LCD\_writeData is known not to modify X so no need to preserve

call LCD\_WriteData ; Write data to screen

inc X ; Advance pointer to next character

jmp .Loop\_PrString ; Go get next character

.End\_LCD\_PrString:

RAM\_EPILOGUE RAM\_USE\_CLASS\_3

ret

.ENDSECTION

.SECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_WriteData

;

; DESCRIPTION:

; Write a byte to the LCD's data register.

;

;-----------------------------------------------------------------------------

;

; ARGUMENTS:

; A contains byte to be written to LCD data register

;

; RETURNS: none

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

; Currently only the page pointer registers listed below are modified:

; CUR\_PP

;

LCD\_WriteData:

\_LCD\_WriteData:

LCD\_Write\_Data: ; Do not use

\_LCD\_Write\_Data: ; Do not use

RAM\_PROLOGUE RAM\_USE\_CLASS\_1

call LCD\_Check\_Ready ; Make sure controller is ready

; A is preserved in LCD\_Check\_Ready

push A ; Save copy of character

asr A ; Shift high nibble to right

asr A

asr A

asr A

and A,0Fh ; Mask off high nibble

call LCD\_WDATA\_Nibble ; Write Upper nibble

pop A ; Retrieve copy of character

and A,0Fh ; Mask off high nibble

nop

nop

nop

call LCD\_WDATA\_Nibble ; Write Lower nibble

RAM\_EPILOGUE RAM\_USE\_CLASS\_1

ret

.ENDSECTION

.SECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_Control

;

; DESCRIPTION:

; Write a byte to the LCD's control register.

;

;-----------------------------------------------------------------------------

;

; ARGUMENTS:

; A contains data to be written to LCD control register.

;

; RETURNS: none

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

; Currently only the page pointer registers listed below are modified:

; CUR\_PP

;

LCD\_Control:

\_LCD\_Control:

RAM\_PROLOGUE RAM\_USE\_CLASS\_1

call LCD\_Check\_Ready ; Make sure controller is ready

; A is preserved in LCD\_Check\_Ready

push A ; Save copy of byte

asr A ; Shift Upper Nibble to right

asr A

asr A

asr A

and A,0Fh ; Mask off, just in case

call LCD\_WCNTL\_Nibble ; Write high nibble

pop A ; Restore copy of byte

and A,0Fh ; Mask off high nibble

nop

nop

nop

call LCD\_WCNTL\_Nibble ; Write Lower nibble

RAM\_EPILOGUE RAM\_USE\_CLASS\_1

ret

.ENDSECTION

.SECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_WCNTL\_Nibble

;

; DESCRIPTION:

; Write a single nibble to the LCD's command register

;

;-----------------------------------------------------------------------------

;

; ARGUMENTS:

; A[3:0] Contains Nibble to be written to command register

;

; RETURNS: none

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

; Currently only the page pointer registers listed below are modified:

; CUR\_PP

;

LCD\_WCNTL\_Nibble:

RAM\_PROLOGUE RAM\_USE\_CLASS\_4

push A

RAM\_SETPAGE\_CUR >Port\_2\_Data\_SHADE ; Set CUR\_PP to LCD variable address

and [Port\_2\_Data\_SHADE],~LCD\_PORT\_MASK

mov A,[Port\_2\_Data\_SHADE]

mov reg[LCD\_Port],A ; Reset control lines

pop A

and A,LCD\_DATA\_MASK ; Make sure no bogus data in MSN

or A,LCD\_E ; Bring "E" Enable line high

or A,[Port\_2\_Data\_SHADE] ; OR in bit 7 just

mov reg[LCD\_Port], A ; Write data

mov [Port\_2\_Data\_SHADE],A ; Keep shadow register in sync

nop

and A,(~LCD\_PORT\_MASK|LCD\_DATA\_MASK) ; Disable E signal and leave data on bus.

mov [Port\_2\_Data\_SHADE],A ; Keep shadow register in sync

mov reg[LCD\_Port],A

RAM\_EPILOGUE RAM\_USE\_CLASS\_4

ret

.ENDSECTION

.SECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_WDATA\_Nibble

;

; DESCRIPTION:

; Write a single nibble to the LCD's DATA register

;

;-----------------------------------------------------------------------------

;

; ARGUMENTS:

; A[3:0] Contains Nibble to be written to data register

;

; RETURNS: none

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

; Currently only the page pointer registers listed below are modified:

; CUR\_PP

;

LCD\_WDATA\_Nibble:

RAM\_PROLOGUE RAM\_USE\_CLASS\_4

push A

RAM\_SETPAGE\_CUR >Port\_2\_Data\_SHADE ; Set CUR\_PP to LCD variable address

and [Port\_2\_Data\_SHADE],~LCD\_PORT\_MASK

or [Port\_2\_Data\_SHADE],LCD\_RS ; Raise RS to signify a Data Write

mov A,[Port\_2\_Data\_SHADE]

mov reg[LCD\_Port],A

pop A

and A,LCD\_DATA\_MASK ; Make sure no bogus data in A[7:4]

or A,(LCD\_E | LCD\_RS) ; Bring "E" Enable line high

or A,[Port\_2\_Data\_SHADE] ; Keep shadow in sync

mov reg[LCD\_Port], A ; Write data

mov [Port\_2\_Data\_SHADE],A ; Keep shadow in sync

NOP

and A,(~LCD\_PORT\_MASK|LCD\_DATA\_MASK|LCD\_RS) ; Disable E signal and leave Data on bus

mov [Port\_2\_Data\_SHADE],A ; keep shadow in sync

mov reg[LCD\_Port],A

RAM\_EPILOGUE RAM\_USE\_CLASS\_4

ret

.ENDSECTION

.SECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_Check\_Ready

;

; DESCRIPTION:

; Wait until LCD has completed last command.

;

;-----------------------------------------------------------------------------

;

; ARGUMENTS: none

;

; RETURNS: none

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

; Currently only the page pointer registers listed below are modified:

; CUR\_PP

;

; If LCD is not present, this routine may never return.

;

LCD\_Check\_Ready:

RAM\_PROLOGUE RAM\_USE\_CLASS\_4

push A ; Save Accumulator

RAM\_SETPAGE\_CUR >Port\_2\_Data\_SHADE ; Set CUR\_PP to LCD variable address

and [Port\_2\_Data\_SHADE],~LCD\_PORT\_MASK ; Mask of all LCD bits

mov A,[Port\_2\_Data\_SHADE]

mov reg[LCD\_Port],A ; Zero LCD port bits

and [Port\_2\_DriveMode\_0\_SHADE],~LCD\_DATA\_MASK ; Clear out LCD mode bits.

mov A,[Port\_2\_DriveMode\_0\_SHADE]

M8C\_SetBank1 ; Change port mode to read status

mov reg[LCD\_PortMode0],A ; Setup LCD Port for reading

M8C\_SetBank0

or [Port\_2\_Data\_SHADE],LCD\_RW ; Raise RW to signify Read operation

mov A,[Port\_2\_Data\_SHADE]

mov reg[LCD\_Port],A

NOP

push X ; Save 'X' register

mov X,255 ; 255 Attempts

.LCD\_RDY\_LOOP:

or [Port\_2\_Data\_SHADE], LCD\_CNTL\_READ ; Raise E to start cycle

mov A,[Port\_2\_Data\_SHADE]

mov reg[LCD\_Port],A

nop ; Wait 2 nops to make sure data is ready

nop

mov A,reg[LCD\_Port]

; The code below is used to work around the async read issue with the ICE with the

; 25/26xxx family of devices. It will help to eliminate "Invalid memory reference"

; errors. It is not required when running without the ICE or when using any other

; family besides the 25/26xxx family. If not using the ICE or with any other family

; the ICE\_PORT\_SYNC flag should be set to 0.

IF(ICE\_PORT\_SYNC)

mov reg[ 0xfa], A

mov A, reg[0xfa]

ENDIF

push A

and [Port\_2\_Data\_SHADE],(~LCD\_PORT\_MASK | LCD\_RW) ; Lower E signal

mov A,[Port\_2\_Data\_SHADE]

mov reg[LCD\_Port],A

nop ; Add delay for the slowest part and the

nop ; fastest PSoC

nop

; Get the LSBs

or [Port\_2\_Data\_SHADE],LCD\_CNTL\_READ ; Raise E to start cycle

mov A,[Port\_2\_Data\_SHADE]

mov reg[LCD\_Port],A

nop

nop

and [Port\_2\_Data\_SHADE],(~LCD\_PORT\_MASK | LCD\_RW) ; Lower E signal

mov A,[Port\_2\_Data\_SHADE]

mov reg[LCD\_Port],A

pop A

and A,LCD\_READY\_BIT ; Check busy

jz .UNLOCK

dec X

jnz .LCD\_RDY\_LOOP ; If LCD still busy, read again for 255 times

.UNLOCK:

pop X ; Restore 'X' register

or [Port\_2\_DriveMode\_0\_SHADE],LCD\_PORT\_WRITE ; Revert Data bit to Write mode

mov A,[Port\_2\_DriveMode\_0\_SHADE]

M8C\_SetBank1

mov reg[LCD\_PortMode0],A ; Setup LCD Port for writing

M8C\_SetBank0

pop A

RAM\_EPILOGUE RAM\_USE\_CLASS\_4 ; Restore Accumulator

ret

.ENDSECTION

.SECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_Start

; FUNCTION NAME: LCD\_Init

;

; DESCRIPTION:

; Initialize LCD

;

;-----------------------------------------------------------------------------

;

; ARGUMENTS: none

;

; RETURNS: none

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

; Currently only the page pointer registers listed below are modified:

; CUR\_PP

;

; THEORY of OPERATION or PROCEDURE:

; REGISTERS ARE VOLATILE: THE A AND X REGISTERS MAY BE MODIFIED!

; This initialization is a bit long, but it should work for

; most 2 and 4 line LCDs.

;

LCD\_Start:

\_LCD\_Start:

LCD\_Init:

\_LCD\_Init:

RAM\_PROLOGUE RAM\_USE\_CLASS\_4

RAM\_SETPAGE\_CUR >Port\_2\_Data\_SHADE ; Set CUR\_PP to LCD variable address

and [Port\_2\_DriveMode\_0\_SHADE],~LCD\_PORT\_MASK ; Mask off LCD bits

or [Port\_2\_DriveMode\_0\_SHADE],LCD\_PORT\_WRITE ; Set LCD port for writing

and [Port\_2\_DriveMode\_1\_SHADE],~LCD\_PORT\_MASK ; Mask off LCD bits

mov A,[Port\_2\_DriveMode\_0\_SHADE]

M8C\_SetBank1

mov reg[LCD\_PortMode0],A ; Setup LCD Port for writing

mov A,[Port\_2\_DriveMode\_1\_SHADE]

mov reg[LCD\_PortMode1],A

M8C\_SetBank0

mov A,250 ; Delay for 12.5 mSec (250 \* 50uSec)

call LCD\_Delay50uTimes

mov A,250 ; Delay for 12.5 mSec (250 \* 50uSec)

call LCD\_Delay50uTimes

mov A,03h

call LCD\_WCNTL\_Nibble

mov A,82 ; Delay for 4.1 mSec (82 \* 50uSec)

call LCD\_Delay50uTimes

mov A,03h

call LCD\_WCNTL\_Nibble

call LCD\_Delay50u

call LCD\_Delay50u

call LCD\_Delay50u

mov A,03h

call LCD\_WCNTL\_Nibble

mov A,90 ; Delay for 4.5 mSec (90 \* 50uSec)

call LCD\_Delay50uTimes

mov A,02h

call LCD\_WCNTL\_Nibble

mov A,90 ; Delay for 4.5 mSec (90 \* 50uSec)

call LCD\_Delay50uTimes

mov A,08h

call LCD\_Control

mov A,90 ; Delay for 4.5 mSec (90 \* 50uSec)

call LCD\_Delay50uTimes

mov A,01h

call LCD\_Control

mov A,90 ; Delay for 4.5 mSec (90 \* 50uSec)

call LCD\_Delay50uTimes

mov A,06h

call LCD\_Control

mov A,0Eh

call LCD\_Control

mov A,LCD\_4BIT\_2LINE ; Setup for 4 bit interface, 2 line

call LCD\_Control

mov A,LCD\_DISP\_OFF

call LCD\_Control

mov A,LCD\_DISP\_ON

call LCD\_Control

mov A,LCD\_DISP\_INC

call LCD\_Control

mov A,90 ; Delay for 4.5 mSec (90 \* 50uSec)

call LCD\_Delay50uTimes

RAM\_EPILOGUE RAM\_USE\_CLASS\_4

ret

.ENDSECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_Position

;

; DESCRIPTION:

; Position Cursor at Row and Col location

;

;-----------------------------------------------------------------------------

;

; ARGUMENTS:

; A => Row 0 to 3

; X => Col 0 to 39+

;

; RETURNS: none

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

.LITERAL

LCD\_ROW\_OFFSET::

DB LCD\_ROW1\_OFFSET, LCD\_ROW2\_OFFSET, LCD\_ROW3\_OFFSET, LCD\_ROW4\_OFFSET

.ENDLITERAL

.SECTION

LCD\_Position:

\_LCD\_Position:

RAM\_PROLOGUE RAM\_USE\_CLASS\_2

and A,03h ; Mask off 2 bits for row address 0 to 3

push X ; Store COL

index LCD\_ROW\_OFFSET ; Get ROW memory offset from table

mov X,SP ; Get Stack pointer

add A,[X+(-1)] ; Add the COL to the display pointer

pop X

call LCD\_Control ; Write control byte

RAM\_EPILOGUE RAM\_USE\_CLASS\_2

ret

.ENDSECTION

.SECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_Delay50uTimes

;

; DESCRIPTION:

; Delay increments of 50uSeconds

;

;-----------------------------------------------------------------------------

;

; ARGUMENTS:

; A contains the delay multiplier

;

; RETURNS:

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

;

LCD\_Delay50uTimes:

\_LCD\_Delay50uTimes:

RAM\_PROLOGUE RAM\_USE\_CLASS\_1

call LCD\_Delay50u

dec A

jnz LCD\_Delay50uTimes

RAM\_EPILOGUE RAM\_USE\_CLASS\_1

ret

.ENDSECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_Delay50u

;

; DESCRIPTION:

; Delay 50uSec for any clock frequency from 1.5MHz to 24MHz

; Slower clock frequencies the delay will be;

; 1.5

; -------------- \* 50uSec

; clock\_freq(MHz)

;

;

;-----------------------------------------------------------------------------

;

; ARGUMENTS: none

;

; RETURNS: none

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

; THEORY of OPERATION or PROCEDURE:

;

.LITERAL

LCD\_Delay50u\_Table::

DB 08h, 19h, 3Ah, 7Ch, 01h, 01h, 01h, 01h

; 3MHz, 6MHz, 12MHz, 24MHz, 1.5MHz, 750kHz, 188kHz, 94kHz

.ENDLITERAL

.SECTION

LCD\_Delay50u:

\_LCD\_Delay50u: ; [11] Call

RAM\_PROLOGUE RAM\_USE\_CLASS\_1

push A

M8C\_SetBank1 ; [4]

mov A, reg[OSC\_CR0] ; [6] Get delay value

M8C\_SetBank0 ; [4]

and A,07h ; [4] Mask off only the clock bits

cmp A,05h

jnc .Delay50u\_End

index LCD\_Delay50u\_Table ; [13] Get delay value

.Delay50u\_Loop: ;

dec A ; [4]

jnz .Delay50u\_Loop ; [5]

.Delay50u\_End:

pop A

RAM\_EPILOGUE RAM\_USE\_CLASS\_1

ret

.ENDSECTION

;-----------------------------------------------------------------------------

; If bargraph is not enabled, the following functions are not required.

;-----------------------------------------------------------------------------

IF (LCD\_BARGRAPH\_ENABLE)

IF SYSTEM\_SMALL\_MEMORY\_MODEL

.SECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_DrawBG

;

; DESCRIPTION:

; This legacy fastcall version are provided only to support existing small

; memory model assembly language code---it does not work in the large memory

; model.

;

; \*\* This legacy fastcall version is provided on a temporary basis to

; \*\* ease the transition to the 4.2 release of PSoC Designer. Its use is

; \*\* deprecated and its status is "No Further Maintenance". If you call this

; \*\* function in assembly you should convert to \_LCD\_DrawVBG

; \*\* (with a leading underscore) and the fastcall16 interface

;

; Draw a horizontal bargraph on the LCD with the given parameters. This

; is a legacy function that is intended to support existing Assembly

; language programs that call this function. This should not be used for

; new code or with Large Memory Model programs.

;-----------------------------------------------------------------------------

;

; LEGACY FASTCALL ARGUMENTS:

; A => Starting row for bargraph 0 to 3

; [X] => Starting Column for bargraph 0 to 39+

; [x-1] => Length of bargraph in chars 1 to 40+

; [X-2] => Position of pointer in segments 5 times Length

;

;

; RETURNS: none

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

; If LCD\_Init is not called before this function, the

; bargraph will not be drawn properly.

;

; Stack offset constants

BG\_COLX: equ 0 ; Stack position of Column

BG\_CHAR\_LENX: equ -1 ; Stack position of Length

BG\_LENGTHX: equ -2 ; Stack position of bargraph pointer position

LCD\_DrawBG:

push X

mov X,[X+BG\_COLX] ; Row in A, Col in X

call LCD\_Position ; Set cursor position

pop X ; Restore pointer

.LCD\_BG\_LOOP1X:

cmp [X+BG\_LENGTHX],00h ; Check for past end of BG

jnz .LCD\_CHECK1X

mov A,00h ; Load empty character

jmp .LCD\_BG\_DOITX ;

.LCD\_CHECK1X:

cmp [X+BG\_LENGTHX],06h ; Check if BG pointer is at this character

jnc .LCD\_CHECK2X ; Note yet, use full character

mov A,[X+BG\_LENGTHX]

sub [X+BG\_LENGTHX],A

jmp .LCD\_BG\_DOITX

.LCD\_CHECK2X: ; Put index to full character

mov A, 06h

sub [X+BG\_LENGTHX],05h ; Subtract another 5 positions

.LCD\_BG\_DOITX:

call LCD\_WriteData ; Display BG character

dec [X+BG\_CHAR\_LENX] ; Dec Char count

jnz .LCD\_BG\_LOOP1X ; Do it all over again

ret

.ENDSECTION

ENDIF ; SYSTEM\_SMALL\_MEMORY\_MODEL

.SECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_DrawBG

;

; DESCRIPTION:

; Draw a horizontal bargraph on the LCD with the given parameters.

;

;

;-----------------------------------------------------------------------------

;

; FASTCALL16 ARGUMENTS:

; [SP-3] => Starting row for bargraph 0 to 3

; [SP-4] => Starting Column for bargraph 0 to 39+

; [SP-5] => Length of bargraph in chars 1 to 40+

; [SP-6] => Position of pointer in segments 5 times Length

;

;

; RETURNS: none

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

; Currently only the page pointer registers listed below are modified:

; CUR\_PP

;

; If LCD\_Init is not called before this function, the

; bargraph will not be drawn properly.

;

; Stack offset constants

BG\_ROW: equ -3

BG\_COL: equ -4 ; Stack position of Column

BG\_CHAR\_LEN: equ -5 ; Stack position of Length

BG\_LENGTH: equ -6 ; Stack position of bargraph pointer position

\_LCD\_DrawBG:

RAM\_PROLOGUE RAM\_USE\_CLASS\_2

mov X, SP

push X

mov A,[X+BG\_ROW] ; Row in A

mov X,[X+BG\_COL] ; Col in X

RAM\_EPILOGUE RAM\_USE\_CLASS\_2

call LCD\_Position ; Set cursor position

RAM\_PROLOGUE RAM\_USE\_CLASS\_2

pop X

.LCD\_BG\_LOOP1:

cmp [X+BG\_LENGTH],00h ; Check for past end of BG

jnz .LCD\_CHECK1

mov A,00h ; Load empty character

jmp .LCD\_BG\_DOIT ;

.LCD\_CHECK1:

cmp [X+BG\_LENGTH],06h ; Check if BG pointer is at this character

jnc .LCD\_CHECK2 ; Note yet, use full character

mov A,[X+BG\_LENGTH]

sub [X+BG\_LENGTH],A

jmp .LCD\_BG\_DOIT

.LCD\_CHECK2: ; Put index to full character

mov A, 06h

sub [X+BG\_LENGTH],05h ; Subtract another 5 positions

.LCD\_BG\_DOIT:

call LCD\_WriteData ; Display BG character

dec [X+BG\_CHAR\_LEN] ; Dec Char count

jnz .LCD\_BG\_LOOP1 ; Do it all over again

RAM\_EPILOGUE RAM\_USE\_CLASS\_2

ret

.ENDSECTION

IF SYSTEM\_SMALL\_MEMORY\_MODEL

.SECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_DrawVBG

;

; DESCRIPTION:

; This legacy fastcall version are provided only to support existing small

; memory model assembly language code---it does not work in the large memory

; model.

;

; \*\* This legacy fastcall version is provided on a temporary basis to

; \*\* ease the transition to the 4.2 release of PSoC Designer. Its use is

; \*\* deprecated and its status is "No Further Maintenance". If you call this

; \*\* function in assembly you should convert to \_LCD\_DrawVBG

; \*\* (with a leading underscore) and the fastcall16 interface

;

; Draw a vertical bargraph on the LCD with the given parameters. This

; is a legacy function that is intended to support existing Assembly

; language programs that call this function. This should not be used for

; new code or with Large Memory Model programs.

;-----------------------------------------------------------------------------

;

; LEGACY FASTCALL ARGUMENTS:

; A => Starting row for bargraph 0 to 3

; [X] => Starting Column for bargraph 0 to 40+

; [x-1] => Height of bargraph in chars 1 - 4

; [X-2] => Position of pointer in segments 8 times height

; RETURNS:

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

; If LCD\_Init is not called before this function, the

; bargraph will not be drawn properly.

;

; Stack offset constants

VBG\_COLX: equ 0

VBG\_CHAR\_HEIGHTX: equ -1

VBG\_SEG\_HEIGHTX: equ -2

LCD\_DrawVBG:

and A,03h ; Make sure only rows 0 - 3 are valid

.VBG\_LOOPX:

push A

index LCD\_ROW\_OFFSET ; Get row offset

add A,[X+VBG\_COLX] ; Add column offset to position

call LCD\_Control ; Position Cursor

cmp [X+VBG\_SEG\_HEIGHTX],00h ; Check for zero segs

jnz .VBG\_NZ\_SEGX

mov A,' ' ; Load space character

jmp .VBG\_WRITE\_CHARX

.VBG\_NZ\_SEGX:

cmp [X+VBG\_SEG\_HEIGHTX],09h ; Check for full segment

jnc .VBG\_FULL\_SEGX

; Partial segment between 1 and 8

mov A,[X+VBG\_SEG\_HEIGHTX]

dec A

mov [X+VBG\_SEG\_HEIGHTX],00h ; Zero segment height

jmp .VBG\_WRITE\_CHARX

.VBG\_FULL\_SEGX: ; Bargaph

sub [X+VBG\_SEG\_HEIGHTX],08h ; Subtract full segment

mov A,07h ; Load full segment

.VBG\_WRITE\_CHARX: ; Write character to display

call LCD\_WriteData ; Write value

pop A

dec A

dec [X+VBG\_CHAR\_HEIGHTX]

jnz .VBG\_LOOPX

ret

.ENDSECTION

ENDIF ; SYSTEM\_SMALL\_MEMORY\_MODEL

.SECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_DrawVBG

;

; DESCRIPTION:

; Draw a vertical bargraph on the LCD with the given parameters.

;

;

;-----------------------------------------------------------------------------

;

; FASTCALL16 ARGUMENTS:

;

; [SP-3] => Starting row for bargraph 0 to 3

; [SP-4] => Starting Column for bargraph 0 to 40+

; [SP-5] => Height of bargraph in chars 1 - 4

; [SP-6] => Position of pointer in segments 8 times height

; RETURNS:

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

; Currently only the page pointer registers listed below are modified:

; CUR\_PP

;

; If LCD\_Init is not called before this function, the

; bargraph will not be drawn properly.

;

; Stack offset constants

VBG\_ROW: equ -3

VBG\_COL: equ -4

VBG\_CHAR\_HEIGHT: equ -5

VBG\_SEG\_HEIGHT: equ -6

\_LCD\_DrawVBG:

RAM\_PROLOGUE RAM\_USE\_CLASS\_2

mov X, SP

mov A, [X+VBG\_ROW]

and A,03h ; Make sure only rows 0 - 3 are valid

.VBG\_LOOP:

push A

index LCD\_ROW\_OFFSET ; Get row offset

add A,[X+VBG\_COL] ; Add column offset to position

call LCD\_Control ; Position Cursor

cmp [X+VBG\_SEG\_HEIGHT],00h ; Check for zero segs

jnz .VBG\_NZ\_SEG

mov A,' ' ; Load space character

jmp .VBG\_WRITE\_CHAR

.VBG\_NZ\_SEG:

cmp [X+VBG\_SEG\_HEIGHT],09h ; Check for full segment

jnc .VBG\_FULL\_SEG

; Partial segment between 1 and 8

mov A,[X+VBG\_SEG\_HEIGHT]

dec A

mov [X+VBG\_SEG\_HEIGHT],00h ; Zero segment height

jmp .VBG\_WRITE\_CHAR

.VBG\_FULL\_SEG: ; Bargaph

sub [X+VBG\_SEG\_HEIGHT],08h ; Subtract full segment

mov A,07h ; Load full segment

.VBG\_WRITE\_CHAR: ; Write character to display

call LCD\_WriteData ; Write value

pop A

dec A

dec [X+VBG\_CHAR\_HEIGHT]

jnz .VBG\_LOOP

RAM\_EPILOGUE RAM\_USE\_CLASS\_2

ret

.ENDSECTION

.SECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_InitVBG

;

; DESCRIPTION:

; Initialize the vertical bargraph characters.

;

;-----------------------------------------------------------------------------

;

; ARGUMENTS: none

;

; RETURNS: none

;

; SIDE EFFECTS:

; REGISTERS ARE VOLATILE: THE A AND X REGISTERS MAY BE MODIFIED!

; Only one type of bargraph (horizontal or vertical) may be used

; at a time since they each require their own set of characters.

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

; Currently only the page pointer registers listed below are modified:

; CUR\_PP

;

; Stack offset constants

VBGDATA\_CTR: equ 00h ; Char data count stack offset

VBG\_BYTES: equ 01h ; Byte counter stack offset

LCD\_InitVBG:

\_LCD\_InitVBG:

RAM\_PROLOGUE RAM\_USE\_CLASS\_2

mov X,SP ; Get location of stack

push A ; Create 2 locations

push A

mov A,LCD\_CG\_RAM\_OFFSET ; Setup pointer

call LCD\_Control ; Position the CG pointer

mov [X+VBGDATA\_CTR],01h ; Reset data counter

.VBG\_Loop1: ; loop once for each 8 characters

mov [X+VBG\_BYTES],08h ; Load cycle pointer

.VBG\_Loop2: ; Loop once for each line in character (8 times)

mov A,[X+VBGDATA\_CTR]

cmp A,[X+VBG\_BYTES]

jnc .VBG\_SOLID

mov A,00h ; Empty line

jmp .VBG\_Load ; Jump to load the bargraph

.VBG\_SOLID:

mov A,FFh ; Load solid line

.VBG\_Load:

call LCD\_WriteData ; character data

dec [X+VBG\_BYTES] ; Dec byte counter

jnz .VBG\_Loop2 ; End Loop 2

inc [X+VBGDATA\_CTR]

cmp [X+VBGDATA\_CTR],09h

jnz .VBG\_Loop1 ; End Loop1

pop A

pop A

mov A,LCD\_DISP\_ON ; Turn on display, don't really

call LCD\_Control ; need this.

RAM\_EPILOGUE RAM\_USE\_CLASS\_2

ret

.ENDSECTION

;-----------------------------------------------------------------------------

; FUNCTION NAME: LCD\_InitBG

;

; DESCRIPTION:

; Initialize horizontal bargraph characters

;

;-----------------------------------------------------------------------------

;

; ARGUMENTS:

; A = type 0 = full |||||||||........

; 1 = single vertical line ..........|......

;

; RETURNS:

;

; SIDE EFFECTS:

; The A and X registers may be modified by this or future implementations

; of this function. The same is true for all RAM page pointer registers in

; the Large Memory Model. When necessary, it is the calling function's

; responsibility to preserve their values across calls to fastcall16

; functions.

;

; Currently only the page pointer registers listed below are modified:

; CUR\_PP

;

; Only one type of bargraph (horizontal or vertical) may be used

; at a time since they each require their own set of characters.

;

; THEORY of OPERATION or PROCEDURE:

; This function writes to the LCD character RAM to generate 8 custom

; characters used to generated one of two horizontal bargraphs.

;

.LITERAL

LCD\_BG\_TYPE1:: ; ....., |...., ||..., |||.., ||||., |||||, |||||

DB 00h, 10h, 18h, 1Ch, 1Eh, 1Fh, 1Fh

LCD\_BG\_TYPE2:: ; ....., |...., .|..., ..|.., ...|., ....|, .....

DB 00h, 10h, 08h, 04h, 02h, 01h, 00h

.ENDLITERAL

.SECTION

; Stack offset constants

BGDATA\_PTR: equ 00h ; Stack offsets

BGCHARS: equ 01h

BGTYPE: equ 02h

LCD\_InitBG:

\_LCD\_InitBG:

RAM\_PROLOGUE RAM\_USE\_CLASS\_2

mov X,SP ; Get location of stack

add SP,3

mov [X+BGTYPE],A ; Store the bargraph type

mov A,LCD\_CG\_RAM\_OFFSET ; Setup pointer

call LCD\_Control ; Position the CG pointer

mov [X+BGDATA\_PTR],00h ; Reset pointer to BG data

.BG\_Loop1:

mov [X+BGCHARS],08h ; Load cycle pointer

.BG\_Loop2:

mov A,[X+BGDATA\_PTR]

cmp [X+BGTYPE],00h ; Check which bargraph

jnz .BG\_OTHER

index LCD\_BG\_TYPE1

jmp .BG\_Load

.BG\_OTHER:

index LCD\_BG\_TYPE2

.BG\_Load:

call LCD\_WriteData

dec [X+BGCHARS] ; Character builder counter

jnz .BG\_Loop2

inc [X+BGDATA\_PTR] ; Advance to next character

cmp [X+BGDATA\_PTR],07h

jnz .BG\_Loop1

add SP,-3

mov A,LCD\_DISP\_ON

call LCD\_Control

RAM\_EPILOGUE RAM\_USE\_CLASS\_2

ret

.ENDSECTION

ENDIF

; End of File LCD.asm