// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// Copyright (C) 2009 Texas Instruments Incorporated - http://www.ti.com/

//

//

// Redistribution and use in source and binary forms, with or without

// modification, are permitted provided that the following conditions

// are met:

//

// Redistributions of source code must retain the above copyright

// notice, this list of conditions and the following disclaimer.

//

// Redistributions in binary form must reproduce the above copyright

// notice, this list of conditions and the following disclaimer in the

// documentation and/or other materials provided with the

// distribution.

//

// Neither the name of Texas Instruments Incorporated nor the names of

// its contributors may be used to endorse or promote products derived

// from this software without specific prior written permission.

//

// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS

// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT

// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR

// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT

// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,

// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT

// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,

// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY

// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT

// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE

// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Stopwatch functions.

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Include section

// system

**#include** "project.h"

**#include** <string.h>

// driver

**#include** "stopwatch.h"

**#include** "ports.h"

**#include** "display.h"

**#include** "timer.h"

**#include** "buzzer.h"

// logic

**#include** "menu.h"

**#include** "alarm.h"

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Prototypes section

**void** **start\_stopwatch**(**void**);

**void** **stop\_stopwatch**(**void**);

**void** **reset\_stopwatch**(**void**);

**void** **stopwatch\_tick**(**void**);

**void** **update\_stopwatch\_timer**(**void**);

**void** **mx\_stopwatch**(u8 line);

**void** **sx\_stopwatch**(u8 line);

**void** **display\_stopwatch**(u8 line, u8 update);

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Defines section

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Global Variable section

**struct** stopwatch sStopwatch;

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Extern section

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// @fn update\_stopwatch\_timer

// @brief Set new compare time for next 1/1Hz or 1/100Hz interrupt. Takes care for exact 1

// second timing.

// @param ticks (1 tick = 1/32768 sec)

// @return none

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void** **update\_stopwatch\_timer**(**void**)

{

u16 value;

// Load CCR register with next capture time

**if** (sStopwatch.viewStyle == DISPLAY\_DEFAULT\_VIEW)

{

// Timer interrupts occur every 32768/100 = 328 ACLK

// --> stopwatch runs too slow (1 sec nominal != 100 interupts \* 328 ACLK = 32800 ACLK =

// 1.00098 sec)

// --> ideally correct timer value every 10 ticks by (32768 - 32800)/10 = 3.2

// --> correct timer value every 10Hz by 3,

// --> correct timer value every 1Hz correct by 5

value = TA0CCR2 + STOPWATCH\_100HZ\_TICK;

**if** (sStopwatch.swtIs1Hz)

{

value -= 5;

sStopwatch.swtIs1Hz = 0;

sStopwatch.swtIs10Hz = 0;

}

**else** **if** (sStopwatch.swtIs10Hz)

{

value -= 3;

sStopwatch.swtIs10Hz = 0;

}

}

**else** // Alternative view

{

// Timer interrupts occur every 32768/1 = 32768 ACLK

value = TA0CCR2 + STOPWATCH\_1HZ\_TICK;

}

// Update CCR

TA0CCR2 = value;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// @fn stopwatch\_tick

// @brief Called by 1/100Hz interrupt handler.

// Increases stopwatch counter and triggers display update.

// @param none

// @return none

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void** **stopwatch\_tick**(**void**)

{

**static** u8 delay = 0;

// Default view (< 20 minutes): display and count MM:SS:hh

**if** (sStopwatch.viewStyle == DISPLAY\_DEFAULT\_VIEW)

{

// Add 1/100 sec

sStopwatch.time[7]++;

// Draw flag minimizes display update activity

//

// swt.drawFlag = 1: second L

// swt.drawFlag = 2: second H/L

// swt.drawFlag = 3: minutes L, second H/L

// swt.drawFlag = 4: minutes H/L, second H/L

// swt.drawFlag = 5: hours L, minutes H/L, second H/L

// swt.drawFlag = 6: hours H/L, minutes H/L, second H/L

// swt.drawFlag = 7: 1/10 sec, 1/100 sec

// swt.drawFlag = 8: 1/100 sec (every 17/100 sec to reduce display draw activity)

**if** (delay++ > 17)

{

sStopwatch.drawFlag = 8;

delay = 0;

}

// Add 1/10 sec

**if** (sStopwatch.time[7] == 0x3A)

{

sStopwatch.time[7] = '0';

sStopwatch.time[6]++;

// 1/10Hz trigger

sStopwatch.swtIs10Hz = 1;

// Update draw flag

sStopwatch.drawFlag = 7;

}

}

**else** // Alternative view (20 minutes .. 20 hours): display and count

// HH:MM:SS

{

// Just add 1 second

sStopwatch.time[6] = 0x3A;

}

// Second overflow?

**if** (sStopwatch.time[6] == 0x3A)

{

// Reset draw flag

sStopwatch.drawFlag = 1;

// 1Hz trigger

sStopwatch.swtIs1Hz = 1;

// Add data

sStopwatch.time[6] = '0';

sStopwatch.time[5]++; // second L (0 - 9)

**if** (sStopwatch.time[5] == 0x3A)

{

sStopwatch.drawFlag++; // 2

sStopwatch.time[5] = '0';

sStopwatch.time[4]++; // second H (0 - 5)

**if** (sStopwatch.time[4] == '6')

{

sStopwatch.drawFlag++; // 3

sStopwatch.time[4] = '0';

sStopwatch.time[3]++; // minutes L (0 - 9)

**if** (sStopwatch.time[3] == 0x3A)

{

sStopwatch.drawFlag++; // 4

sStopwatch.time[3] = '0';

sStopwatch.time[2]++; // minutes H (0 - 5)

**if** (sStopwatch.time[2] == '1')

{

stop\_stopwatch();

sBuzzer.state = BUZZER\_ON\_OUTPUT\_ENABLED; //Does not work, alarm w/ counts?

sStopwatch.state = STOPWATCH\_RUN;

display\_stopwatch(LINE2, DISPLAY\_LINE\_UPDATE\_FULL);

}

}

}

}

}

// Always set display update flag

display.flag.update\_stopwatch = 1;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// @fn reset\_stopwatch

// @brief Clears stopwatch counter and sets stopwatch state to off.

// @param none

// @return none

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void** **reset\_stopwatch**(**void**)

{

// Clear counter

**memcpy**(sStopwatch.time, "00000000", **sizeof**(sStopwatch.time));

// Clear trigger

sStopwatch.swtIs10Hz = 0; // 1/10Hz trigger

sStopwatch.swtIs1Hz = 0; // 1Hz trigger

// Init stopwatch state 'Off'

sStopwatch.state = STOPWATCH\_STOP;

// Default display style is MM:SS:HH

sStopwatch.viewStyle = DISPLAY\_DEFAULT\_VIEW;

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// @fn is\_stopwatch

// @brief Is stopwatch operating and visible?

// @param none

// @return 1=STOPWATCH\_RUN, 0=other states

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

u8 **is\_stopwatch**(**void**)

{

**return** ((sStopwatch.state == STOPWATCH\_RUN) && (ptrMenu\_L2 == &menu\_L2\_Stopwatch));

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// @fn start\_stopwatch

// @brief Starts stopwatch timer interrupt and sets stopwatch state to on.

// @param none

// @return none

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void** **start\_stopwatch**(**void**)

{

// Set stopwatch run flag

sStopwatch.state = STOPWATCH\_RUN;

// Init CCR register with current time

TA0CCR2 = TA0R;

// Load CCR register with next capture time

update\_stopwatch\_timer();

// Reset IRQ flag

TA0CCTL2 &= ~CCIFG;

// Enable timer interrupt

TA0CCTL2 |= CCIE;

// Set stopwatch icon

display\_symbol(LCD\_ICON\_STOPWATCH, SEG\_ON);

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// @fn stop\_stopwatch

// @brief Stops stopwatch timer interrupt and sets stopwatch state to off.

// Does not reset stopwatch count.

// @param none

// @return none

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void** **stop\_stopwatch**(**void**)

{

// Clear timer interrupt enable

TA0CCTL2 &= ~CCIE;

// Clear stopwatch run flag

sStopwatch.state = STOPWATCH\_STOP;

// Clear stopwatch icon

display\_symbol(LCD\_ICON\_STOPWATCH, SEG\_OFF);

// Call draw routine immediately

display\_stopwatch(LINE2, DISPLAY\_LINE\_UPDATE\_FULL);

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// @fn mx\_stopwatch

// @brief Stopwatch set routine. Mx stops stopwatch and resets count.

// @param u8 line LINE2

// @return none

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void** **mx\_stopwatch**(u8 line)

{

// Stop stopwatch

stop\_stopwatch();

// Reset stopwatch count

reset\_stopwatch();

// Display "00:00:00"

display\_stopwatch(line, DISPLAY\_LINE\_UPDATE\_FULL);

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// @fn sx\_stopwatch

// @brief Stopwatch direct function. Button DOWN starts/stops stopwatch, but does not reset

// count.

// @param u8 line LINE2

// @return none

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void** **sx\_stopwatch**(u8 line)

{

// DOWN: RUN, STOP

**if** (button.flag.down)

{

**if** (sStopwatch.state == STOPWATCH\_STOP)

{

// (Re)start stopwatch

start\_stopwatch();

}

**else**

{

// Stop stopwatch

stop\_stopwatch();

}

}

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// @fn display\_stopwatch

// @brief Stopwatch user routine. Sx starts/stops stopwatch, but does not reset count.

// @param u8 line LINE2

// u8 update DISPLAY\_LINE\_UPDATE\_PARTIAL,

// DISPLAY\_LINE\_UPDATE\_FULL

// @return none

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void** **display\_stopwatch**(u8 line, u8 update)

{

// Partial line update only

**if** (update == DISPLAY\_LINE\_UPDATE\_PARTIAL)

{

**if** (display.flag.update\_stopwatch)

{

**if** (sStopwatch.viewStyle == DISPLAY\_DEFAULT\_VIEW)

{

// Display MM:SS:hh

// Check draw flag to minimize workload

**if** (sStopwatch.drawFlag != 0)

{

**switch** (sStopwatch.drawFlag)

{

**case** 4:

display\_char(LCD\_SEG\_L2\_5, sStopwatch.time[2], SEG\_ON);

**case** 3:

display\_char(LCD\_SEG\_L2\_4, sStopwatch.time[3], SEG\_ON);

**case** 2:

display\_char(LCD\_SEG\_L2\_3, sStopwatch.time[4], SEG\_ON);

**case** 1:

display\_char(LCD\_SEG\_L2\_2, sStopwatch.time[5], SEG\_ON);

**case** 7:

display\_char(LCD\_SEG\_L2\_1, sStopwatch.time[6], SEG\_ON);

**case** 8:

display\_char(LCD\_SEG\_L2\_0, sStopwatch.time[7], SEG\_ON);

}

}

}

**else** // DISPLAY\_ALTERNATIVE\_VIEW

{

// Display HH:MM:SS

**switch** (sStopwatch.drawFlag)

{

**case** 6:

display\_char(LCD\_SEG\_L2\_5, sStopwatch.time[0], SEG\_ON);

**case** 5:

display\_char(LCD\_SEG\_L2\_4, sStopwatch.time[1], SEG\_ON);

**case** 4:

display\_char(LCD\_SEG\_L2\_3, sStopwatch.time[2], SEG\_ON);

**case** 3:

display\_char(LCD\_SEG\_L2\_2, sStopwatch.time[3], SEG\_ON);

**case** 2:

display\_char(LCD\_SEG\_L2\_1, sStopwatch.time[4], SEG\_ON);

**case** 1:

display\_char(LCD\_SEG\_L2\_0, sStopwatch.time[5], SEG\_ON);

}

}

}

}

// Redraw whole line

**else** **if** (update == DISPLAY\_LINE\_UPDATE\_FULL)

{

**if** (sStopwatch.viewStyle == DISPLAY\_DEFAULT\_VIEW)

{

// Display MM:SS:hh

display\_chars(LCD\_SEG\_L2\_5\_0, sStopwatch.time + 2, SEG\_ON);

}

**else** // DISPLAY\_ALTERNATIVE\_VIEW

{

// Display HH:MM:SS

display\_chars(LCD\_SEG\_L2\_5\_0, sStopwatch.time, SEG\_ON);

}

display\_symbol(LCD\_SEG\_L2\_COL1, SEG\_ON);

display\_symbol(LCD\_SEG\_L2\_COL0, SEG\_ON);

}

**else** **if** (update == DISPLAY\_LINE\_CLEAR)

{

// Clean up symbols when leaving function

}

}