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

//

// 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.

//

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

// Time functions.

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

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

// Include section

// system

**#include** "project.h"

// driver

**#include** "ports.h"

**#include** "display.h"

**#include** "timer.h"

**#include** "buzzer.h"

// logic

**#include** "alarm.h"

**#include** "menu.h"

**#include** "clock.h"

**#include** "user.h"

**#include** "bluerobin.h"

**#include** "date.h"

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

// Prototypes section

**void** **reset\_clock**(**void**);

**void** **clock\_tick**(**void**);

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

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

**void** **calc\_24H\_to\_12H**(u8 \* hours, u8 \* timeAM);

**void** **conv\_24H\_to\_12H**(u8 \* hours24, u8 \* hours12, u8 \* timeAMorPM);

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

// Defines section

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

// Global Variable section

**struct** time sTime;

// Display values for time format selection

**const** u8 selection\_Timeformat[][4] = {

"24H", "12H"

};

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

// Extern section

**extern** **void** (\*fptr\_lcd\_function\_line1)(u8 line, u8 update);

**extern** **void** (\*fptr\_lcd\_function\_line2)(u8 line, u8 update);

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

// @fn reset\_clock

// @brief Resets clock time to 00:00:00, 24H time format.

// @param none

// @return none

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

**void** **reset\_clock**(**void**)

{

// Set global system time to 0

sTime.system\_time = 0;

// Set main 24H time to start value. 600 counts in 10 minutes.

sTime.hour = 6;

sTime.minute = 00;

sTime.second = 0;

// Display style of both lines is default (HH:MM)

sTime.line1ViewStyle = DISPLAY\_DEFAULT\_VIEW;

// Reset timeout detection

sTime.last\_activity = 0;

}

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

// @fn clock\_tick

// @brief Add 1 second to system time and to display time

// @param none

// @return none

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

**void** **clock\_tick**(**void**)

{

// Use sTime.drawFlag to minimize display updates

// sTime.drawFlag = 1: second

// sTime.drawFlag = 2: minute, second

// sTime.drawFlag = 3: hour, minute

sTime.drawFlag = 1;

// Increase global system time

sTime.system\_time++;

// Add 1 second.

sTime.second++;

// Minus 1 minute.

**if** (sTime.second == 1) //For some reason starts at 6:55 instead of 6:00?

{

sTime.second = 0;

sTime.minute--;

sTime.drawFlag++;

// Minus 1 hour.

**if** (sTime.minute == 0)

{

sTime.minute = 99;

sTime.hour--;

sTime.drawFlag++;

// Turn alarm on and stop timer.

**if** (sTime.hour == 0)

{

sTime.minute = 0;

sAlarm.state = ALARM\_ON; //Does not seem to be working

Timer0\_Stop(); //Find code to stop alarm after x time

}

}

}

}

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

// @fn convert\_hour\_to\_12H\_format

// @brief Convert internal 24H time to 12H time.

// @param u8 hour Hour in 24H format

// @return u8 Hour in 12H format

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

u8 **convert\_hour\_to\_12H\_format**(u8 hour)

{

// 00:00 .. 11:59 --> AM 12:00 .. 11:59

**if** (hour == 0)

**return** (hour + 12);

**else** **if** (hour <= 12)

**return** (hour);

// 13:00 .. 23:59 --> PM 01:00 .. 11:59

**else**

**return** (hour - 12);

}

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

// @fn is\_hour\_am

// @brief Checks if internal 24H time is AM or PM

// @param u8 hour Hour in 24H format

// @return u8 1 = AM, 0 = PM

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

u8 **is\_hour\_am**(u8 hour)

{

// 00:00 .. 11:59 --> AM 12:00 .. 11:59

**if** (hour < 12)

**return** (1);

// 12:00 .. 23:59 --> PM 12:00 .. 11:59

**else**

**return** (0);

}

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

// @fn display\_selection\_Timeformat

// @brief Display time format 12H / 24H.

// @param u8 segments Target segments where to display information

// u32 index 0 or 1, index for value string

// u8 digits Not used

// u8 blanks Not used

// @return none

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

**void** **display\_selection\_Timeformat1**(u8 segments, u32 index, u8 digits, u8 blanks)

{

**if** (index < 2)

display\_chars(segments, (u8 \*) selection\_Timeformat[index], SEG\_ON\_BLINK\_ON);

}

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

// @fn mx\_time

// @brief Clock set routine.

// @param u8 line LINE1, LINE2

// @return none

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

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

{

u8 select;

s32 timeformat;

s16 timeformat1;

s32 hours;

s32 minutes;

s32 seconds;

u8 \*str;

// Clear display

clear\_display\_all();

// Convert global time to local variables

// Global time keeps on ticking in background until it is overwritten

**if** (sys.flag.use\_metric\_units)

{

timeformat = TIMEFORMAT\_24H;

}

**else**

{

timeformat = TIMEFORMAT\_12H;

}

timeformat1 = timeformat;

hours = sTime.hour;

minutes = sTime.minute;

seconds = sTime.second;

// Init value index

select = 0;

// Loop values until all are set or user breaks set

**while** (1)

{

// Idle timeout: exit without saving

**if** (sys.flag.idle\_timeout)

{

// Roll back time format

**if** (timeformat1 == TIMEFORMAT\_24H)

sys.flag.use\_metric\_units = 1;

**else**

sys.flag.use\_metric\_units = 0;

display\_symbol(LCD\_SYMB\_AM, SEG\_OFF);

**break**;

}

// Button STAR (short): save, then exit

**if** (button.flag.star)

{

// Stop clock timer

Timer0\_Stop();

// Store local variables in global clock time

sTime.hour = hours;

sTime.minute = minutes;

sTime.second = seconds;

// Start clock timer

Timer0\_Start();

// Full display update is done when returning from function

display\_symbol(LCD\_SYMB\_AM, SEG\_OFF);

**break**;

}

**switch** (select)

{

**case** 0: // Clear LINE1 and LINE2 and AM icon - required when coming back from

// set\_value(seconds)

clear\_display();

display\_symbol(LCD\_SYMB\_AM, SEG\_OFF);

// Set 24H / 12H time format

set\_value(

&timeformat, 1, 0, 0, 1, SETVALUE\_ROLLOVER\_VALUE + SETVALUE\_DISPLAY\_SELECTION +

SETVALUE\_NEXT\_VALUE, LCD\_SEG\_L1\_3\_1, display\_selection\_Timeformat1);

// Modify global time format variable immediately to update AM/PM icon correctly

**if** (timeformat == TIMEFORMAT\_24H)

sys.flag.use\_metric\_units = 1;

**else**

sys.flag.use\_metric\_units = 0;

select = 1;

**break**;

**case** 1: // Display HH:MM (LINE1) and .SS (LINE2)

str = int\_to\_array(hours, 1, 0);

display\_chars(LCD\_SEG\_L1\_3\_2, str, SEG\_ON);

// display\_symbol(LCD\_SEG\_L1\_COL, SEG\_ON);

str = int\_to\_array(minutes, 2, 0);

display\_chars(LCD\_SEG\_L1\_1\_0, str, SEG\_ON);

str = int\_to\_array(seconds, 2, 0);

display\_chars(LCD\_SEG\_L2\_1\_0, str, SEG\_ON);

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

// Set hours

set\_value(&hours, 2, 0, 0, 23, SETVALUE\_ROLLOVER\_VALUE + SETVALUE\_DISPLAY\_VALUE +

SETVALUE\_NEXT\_VALUE, LCD\_SEG\_L1\_3\_2,

display\_hours);

select = 2;

**break**;

**case** 2: // Set minutes

set\_value(&minutes, 2, 0, 0, 59, SETVALUE\_ROLLOVER\_VALUE + SETVALUE\_DISPLAY\_VALUE +

SETVALUE\_NEXT\_VALUE, LCD\_SEG\_L1\_1\_0,

display\_value);

select = 3;

**break**;

**case** 3: // Set seconds

set\_value(&seconds, 2, 0, 0, 59, SETVALUE\_ROLLOVER\_VALUE + SETVALUE\_DISPLAY\_VALUE +

SETVALUE\_NEXT\_VALUE, LCD\_SEG\_L2\_1\_0,

display\_value);

select = 0;

**break**;

}

}

// Clear button flags

button.all\_flags = 0;

}

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

// @fn sx\_time

// @brief Time user routine. Toggles view style between HH:MM and SS.

// @param line LINE1

// @return none

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

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

{

// Toggle display view style

**if** (sTime.line1ViewStyle == DISPLAY\_DEFAULT\_VIEW)

sTime.line1ViewStyle = DISPLAY\_ALTERNATIVE\_VIEW;

**else**

sTime.line1ViewStyle = DISPLAY\_DEFAULT\_VIEW;

}

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

// @fn display\_time

// @brief Clock display routine. Supports 24H and 12H time format.

// @param u8 line LINE1

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

// DISPLAY\_LINE\_UPDATE\_PARTIAL

// @return none

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

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

{

u8 hour12;

// Partial update

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

{

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

{

**if** (sTime.line1ViewStyle == DISPLAY\_DEFAULT\_VIEW)

{

**switch** (sTime.drawFlag)

{

**case** 3:

**if** (sys.flag.use\_metric\_units)

{

// Display 24H time "HH"

display\_chars(switch\_seg(line, LCD\_SEG\_L1\_3\_2,

LCD\_SEG\_L2\_3\_2), int\_to\_array(sTime.hour, 2,

0), SEG\_ON);

}

**else**

{

// Display 12H time "HH" + AM/PM

hour12 = convert\_hour\_to\_12H\_format(sTime.hour);

display\_chars(switch\_seg(line, LCD\_SEG\_L1\_3\_2,

LCD\_SEG\_L2\_3\_2), int\_to\_array(hour12, 2,

0), SEG\_ON);

display\_am\_pm\_symbol(sTime.hour);

}

**case** 2:

display\_chars(switch\_seg(line, LCD\_SEG\_L1\_1\_0,

LCD\_SEG\_L2\_1\_0), int\_to\_array(sTime.minute, 2,

0), SEG\_ON);

}

}

**else**

{

// Seconds are always updated

display\_chars(switch\_seg(line, LCD\_SEG\_L1\_1\_0,

LCD\_SEG\_L2\_1\_0), int\_to\_array(sTime.second, 2, 0), SEG\_ON);

}

}

}

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

{

// Full update

**if** (sTime.line1ViewStyle == DISPLAY\_DEFAULT\_VIEW)

{

// Display 24H/12H time

**if** (sys.flag.use\_metric\_units)

{

// Display 24H time "HH"

display\_chars(switch\_seg(line, LCD\_SEG\_L1\_3\_2,

LCD\_SEG\_L2\_3\_2), int\_to\_array(sTime.hour, 2, 0), SEG\_ON);

}

**else**

{

// Display 12H time "HH" + AM/PM information

hour12 = convert\_hour\_to\_12H\_format(sTime.hour);

display\_chars(switch\_seg(line, LCD\_SEG\_L1\_3\_2,

LCD\_SEG\_L2\_3\_2), int\_to\_array(hour12, 2, 0), SEG\_ON);

// Display AM/PM information

**if** (line == LINE1)

{

display\_am\_pm\_symbol(sTime.hour);

}

}

// Display minute

display\_chars(switch\_seg(line, LCD\_SEG\_L1\_1\_0,

LCD\_SEG\_L2\_1\_0), int\_to\_array(sTime.minute, 2, 0), SEG\_ON);

display\_symbol(switch\_seg(line, LCD\_SEG\_L1\_COL, LCD\_SEG\_L2\_COL0), SEG\_ON\_BLINK\_ON);

}

**else**

{

// Display seconds

display\_chars(switch\_seg(line, LCD\_SEG\_L1\_1\_0,

LCD\_SEG\_L2\_1\_0), int\_to\_array(sTime.second, 2, 0), SEG\_ON);

display\_symbol(switch\_seg(line, LCD\_SEG\_L1\_DP1, LCD\_SEG\_L2\_DP), SEG\_ON);

}

}

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

{

display\_symbol(switch\_seg(line, LCD\_SEG\_L1\_COL, LCD\_SEG\_L2\_COL0), SEG\_OFF\_BLINK\_OFF);

// Change display style to default (HH:MM)

sTime.line1ViewStyle = DISPLAY\_DEFAULT\_VIEW;

// Clean up AM/PM icon

display\_symbol(LCD\_SYMB\_AM, SEG\_OFF);

}

}