Circuit Health Status Controller

1.1 hw rev. 4

Generated by Doxygen 1.8.6

Sun Nov 8 2015 19:03:31

Contents

Chapter 1

Main Page

This firmware is for the board circuit revision 4 + LCD Alpha shift register display boardThis is the firmware sketch provided by default with the board. Users can modify parameters, strings and I/O behavior depending on their needs and the installation options.

This board runs as an external device able to control - and eventually reset of power on/off your main circuit when some desired or critical / alarming considtions are detected through the sensors.

Note

The most relevant constants and parameters, subject to modification by the users to optimize and customize the Controller Board behaviour are mentioned in the documentation of the components.
The mentioned server example settings can be changed to adapt the Controller behaviour to any kind of device or circuit.

For further details on the board circuit and behaviour see the following article on **`Element14 Arduino blog`**

The board is available on **`Drobott.com`**

For the last updated version clone the repository on **`GitHub`**

Author

Enrico Miglino `baleariddynamics@gmail.com`

Date

November 2015

Version

1.1 (see the Version.h include file for the build and version details), hardware version 1.0 revision 4

Copyright

Chapter 2

Todo List

Member checkPushReleaseButton (int btn)

Optimize ths method with a more consistent series of samples.

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

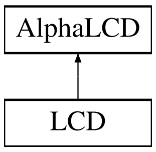
Class Documentation

6.1 LCD Class Reference

Manages the Alphanumeric display for program output messages.

`#include "LCD.h"`

Inheritance diagram for LCD:



Public Member Functions

- LCD ()
- ∼LCD ()
- void enable (bool s)

    Set the display on or off.
- void blink (bool set)

    Set blink mode.
- void error (String m)

    shows an error message
- void error (String m, int x, int y)

    shows an error message at specified coordinates
- void message (String m)

    shows a string message
- void message (String m, int x, int y)

    shows a string message at specified coordinates
- void clean ()

    clean the LCD screen
- void welcome ()

    shows the welcome message
- void showFan (int dFan)
- void showTemp (int dTemp)
- void initFanTemp ()
- void showReset ()

- void showPowerOn ()
- void showPowerOff ()
- void showAction ()
- void initUptime ()
- void showServerStartingStopping ()

Private Member Functions

- LCD (const LCD &c)
- LCD & operator= (const LCD &c)

Private Attributes

- AlphaLCD lcd

    AlphaLCD class inherited instance.

6.1.1 Detailed Description

Manages the Alphanumeric display for program output messages.

This class implements the *AlphaLCD* class that manages the Alphanumeric LCD display hardware using three digital Arduino pins via a shift-out register.

Definition at line 65 of file LCD.h.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 LCD::LCD ( )

6.1.2.2 LCD::∼LCD ( )

6.1.2.3 LCD::LCD (const LCD & *c*) [private]

6.1.3 Member Function Documentation

6.1.3.1 void LCD::blink (bool *set*)

Set blink mode.

6.1.3.2 void LCD::clean ( )

clean the LCD screen

6.1.3.3 void LCD::enable (bool *s*)

Set the display on or off.

6.1.3.4 void LCD::error (String *m*)

shows an error message

6.1.3.5 void LCD::error (String *m*, int *x*, int *y*)

shows an error message at specified coordinates

6.1.3.6 void LCD::initFanTemp ( )

6.1.3.7 void LCD::initUptime ( )

6.1.3.8 void LCD::message (String *m*)

shows a string message

6.1.3.9 void LCD::message (String *m*, int *x*, int *y*)

shows a string message at specified coordinates

6.1.3.10 LCD& LCD::operator= (const LCD & *c*) `[private]`

6.1.3.11 void LCD::showAction ( )

6.1.3.12 void LCD::showFan (int *dFan*)

6.1.3.13 void LCD::showPowerOff ( )

6.1.3.14 void LCD::showPowerOn ( )

6.1.3.15 void LCD::showReset ( )

6.1.3.16 void LCD::showServerStartingStopping ( )

6.1.3.17 void LCD::showTemp (int *dTemp*)

6.1.3.18 void LCD::welcome ( )

shows the welcome message

6.1.4 Member Data Documentation

6.1.4.1 AlphaLCD LCD::lcd `[private]`

AlphaLCD class inherited instance.

Definition at line 70 of file LCD.h.

The documentation for this class was generated from the following file:

- /Volumes/John Doe/Development Projects/Circuit_Health_Controller/CHC/CircuitHealthStatus_Controller-Board/LCD.h

6.2 tmElements_t Struct Reference

Structure defining the used millis() converted values in the proper format.

`#include "UpTime.h"`

Public Attributes

- uint8_t Second
- uint8_t Minute
- uint8_t Hour
- uint8_t Day

6.2.1 Detailed Description

Structure defining the used millis() converted values in the proper format.

Definition at line 25 of file UpTime.h.

6.2.2 Member Data Documentation

6.2.2.1 uint8_t tmElements_t::Day

Definition at line 29 of file UpTime.h.

6.2.2.2 uint8_t tmElements_t::Hour

Definition at line 28 of file UpTime.h.

6.2.2.3 uint8_t tmElements_t::Minute

Definition at line 27 of file UpTime.h.

6.2.2.4 uint8_t tmElements_t::Second

Definition at line 26 of file UpTime.h.

The documentation for this struct was generated from the following file:

- /Volumes/John Doe/Development Projects/Circuit_Health_Controller/CHC/CircuitHealthStatus_Controller-Board/UpTime.h

Chapter 7

File Documentation

7.1 /Volumes/John Doe/Development Projects/Circuit_Health_Controller/CHC/CircuitHealthStatus_ControllerBoard/CircuitHealthStatus_ControllerBoard.ino File Reference

Firmware for the Circuit Health Controller board.

```
#include <AlphaLCD.h>
#include <Streaming.h>
#include "UpTime.h"
#include "LCD.h"
#include "Strings.h"
#include "Version.h"
```

Macros

- #define FANSPEED_MIN 60

  Minimum PWM frequency to start fan Set this value to a value not less than 60 to avoid the motor not starting.
- #define FANSPEED_MAX 255

  Maximum PWM frequency to reach This is the max PWM frequency value. it is not needed to change it.
- #define TEMP_MIN 30

  Minimum temperature to start fan The temperature calculation is based on a case of about 15x15x15 cm internal size.
- #define TEMP_MAX 60

  Maximum temperature before overheating error If the internal temperature reach this level the board enter in a over-heating risk. Leaving the system working for long time at high temperatures may produce serious damage to the components.
- #define UPDATE_FREQ 10

  Loop update frequency This value is a timing delay at the end of every cycle in the loop() function.
- #define FAN_TEST_MS 2500

  Fan full speed test during initialisation This value has no influence on the system control. Just to check if the fan is fully operational (at maximum speed) when the board is powered on.
- #define PWM_FAN 3

  PWM Pin conntrolling the fan speed.
- #define ANALOG_TEMP 0

  Analog pin controlling the temperature.
- #define RESET_BUTTON 2

  Reset button pin.

- #define POWER_BUTTON 7

  Power on/off button pin.
- #define VIBRATION_SENSOR 0

  Vibration sensor pin.
- #define SERVER_POWER_BTN 8

  Power control simulated button pin.
- #define SERVER_RESET_BTN 9

  Reset control simulated button pin.
- #define SERVER_ON 1

  Server powered on.
- #define SERVER_OFF 2

  Server powered off.
- #define SERVER_RESET 3

  Server restarting after reset.
- #define SERVER_POWER_TIME 10000

  msec for server going up (power On and Reset)
- #define SERVER_POWEROFF_TIME 5000

  msec for server goind down
- #define SERVER_RESET_TIME 5000

  msec for server goind down
- #define BUTTON_PRESS_TIMEOUT 5000

  msec Timeout when a button remain pressed
- #define POWER_ON_DELAY 5000

  Power on message delay before starting server.
- #define ALARM_TIMEOUT 5000

  If shock alarm is longer, the system is shutdown.
- #define FIRST_SHOCK_DELAY 1000

  ms before checking the alaram persistance
- #define SENSOR_READINGS 500

  Number of vibration sensors reading for persistance check.
- #define PRESS_POWER_FIRST 1

  Power button has been pressed.
- #define PRESS_POWER_SECOND 2

  Power button pressed again to confirm (Only for shutdown)
- #define PRESS_RESET_FIRST 3
- #define PRESS_RESET_SECOND 4

  Reset button pressed again to confirm.
- #define BUTTON_PRESS_NONE 0

  No buttons has been pressed.
- #define MIN_FANSPEED_PERC 10

  Minimum fan speed PWM percentage to start the fan motor.
- #define RESET_CYCLE_DURATION 500

  ms the simulated server reset button should remain pressed
- #define POWERON_CYCLE_DURATION 500

  ms the simulated server power button should remain pressed to power on
- #define POWEROFF_CYCLE_DURATION 5000

  ms the simulated server power button should remain pressed to power off

Functions

- AlphaLCD lcd (4, 5, 6)

    LCD alphanumeric display class instance Data, latch and clock pins depends on the LCD board connection.
- void setup ()

    Setup method on power-on.
- void loop (void)

    Main loop application.
- void execServerReset ()

    Send a reset signal sequence to the server.
- void execServerPowerOn ()

    Send a power on signal sequence to the server.
- void execServerPowerOff ()

    Send a reset signal sequence to the server.
- int checkPushReleaseButton (int btn)

    Avoid the user keeping the button pressed.
- void checkHealthStatus ()

    Check the health status of the system and update the display.
- int readTemp ()

    Read the analog value from the LM35 temperature sensor with the correction constant to convert in Celsius.
- void showAction (int btn)

    Notification while a user action is active through a button press sequence.
- void welcome ()

    Welcome message shown at device power-on.
- void updateTime ()

    Update the uptime string.
- void initUptime ()

    Initialise the Uptime string.
- void showShock ()

    Show the shock risk string.
- void showReset ()

    Show the reset strings.
- void showPowerOn ()

    Show the powerOn strings.
- void showPowerOff ()

    Show the powerOff strings.
- void showServerStartingStopping ()

    Show the server starting message.
- void initFanTemp ()

    Initialize the temperature and fan fixed text.
- void testFan ()

    Fan fixed text.
- void showFan (int dFan)

    Update the display fan speed (in percentage)
- void showTemp (int dTemp)

    Update the display temperature.
- void message (String m)

    Display a string on the LCD at the cursor position.
- void error (String m, int x, int y)

    Display an error message at the specified cursor coordinates.
- void error (String m)

Display an error message at the cursor position.
- void message (String m, int x, int y)

    Display a string on the LCD at the specified cursor coordinates.
- void clean ()

    Clean the display.

Variables

- int buttonPressed

    the current button pressed
- int serverStatus

    the current status of the server
- int temp

    the current temperature value
- int fanSpeed

    the current fan speed
- int fanSpeedPerc

    the fan speed in percentage (for visualisation)
- int prevFanSpeedPerc

    Last fan speed percentage.
- int prevTemp

    Last temperature read.
- unsigned long shockAlarmTimeout

    The alarm timeout counter.
- unsigned long startTimeSec

    Time value for button press validity telay.

7.1.1 Detailed Description

Firmware for the Circuit Health Controller board. Main sketch file

Definition in file CircuitHealthStatus_ControllerBoard.ino.

7.1.2 Macro Definition Documentation

7.1.2.1 #define ALARM_TIMEOUT 5000

If shock alarm is longer, the system is shutdown.

Definition at line 105 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop().

7.1.2.2 #define ANALOG_TEMP 0

Analog pin controlling the temperature.

Warning

This value is hardwired and should not be changed!

Definition at line 76 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by setup().

7.1.2.3 #define BUTTON_PRESS_NONE 0

No buttons has been pressed.

Definition at line 112 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by checkHealthStatus(), loop(), and setup().

7.1.2.4 #define BUTTON_PRESS_TIMEOUT 5000

msec Timeout when a button remain pressed

Definition at line 103 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop().

7.1.2.5 #define FAN_TEST_MS 2500

Fan full speed test during initialisation This value has no influence on the system control. Just to check if the fan is fully operational (at maximum speed) when the board is powered on.

Definition at line 69 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by testFan().

7.1.2.6 #define FANSPEED_MAX 255

Maximum PWM frequency to reach This is the max PWM frequency value. it is not needed to change it.

Definition at line 49 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by checkHealthStatus(), and testFan().

7.1.2.7 #define FANSPEED_MIN 60

Minimum PWM frequency to start fan Set this value to a value not less than 60 to avoid the motor not starting.

Note

> This parameter value is preset based on a 60 cm diameter fan 12V powered. Different power motors can require an higher frequency to start

Definition at line 45 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by checkHealthStatus(), and testFan().

7.1.2.8 #define FIRST_SHOCK_DELAY 1000

ms before checking the alaram persistance

Definition at line 106 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop().

7.1.2.9 #define MIN_FANSPEED_PERC 10

Minimum fan speed PWM percentage to start the fan motor.

Definition at line 113 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by checkHealthStatus().

7.1.2.10 #define POWER_BUTTON 7

Power on/off button pin.

Warning

This value is hardwired and should not be changed!

Definition at line 82 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop(), setup(), and showAction().

7.1.2.11 #define POWER_ON_DELAY 5000

Power on message delay before starting server.

Definition at line 104 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop().

7.1.2.12 #define POWEROFF_CYCLE_DURATION 5000

ms the simulated server power button should remain pressed to power off

Definition at line 116 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by execServerPowerOff().

7.1.2.13 #define POWERON_CYCLE_DURATION 500

ms the simulated server power button should remain pressed to power on

Definition at line 115 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by execServerPowerOn().

7.1.2.14 #define PRESS_POWER_FIRST 1

Power button has been pressed.

Definition at line 108 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop().

7.1.2.15 #define PRESS_POWER_SECOND 2

Power button pressed again to confirm (Only for shutdown)

Definition at line 109 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop().

7.1.2.16 #define PRESS_RESET_FIRST 3

Definition at line 110 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop().

7.1.2.17 #define PRESS_RESET_SECOND 4

Reset button pressed again to confirm.

Definition at line 111 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop().

7.1.2.18 #define PWM_FAN 3

PWM Pin conntrolling the fan speed.

Warning

This value is hardwired and should not be changed!

Definition at line 73 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by checkHealthStatus(), setup(), and testFan().

7.1.2.19 #define RESET_BUTTON 2

Reset button pin.

Warning

This value is hardwired and should not be changed!

Definition at line 79 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop(), setup(), and showAction().

7.1.2.20 #define RESET_CYCLE_DURATION 500

ms the simulated server reset button should remain pressed

Definition at line 114 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by execServerReset().

7.1.2.21 #define SENSOR_READINGS 500

Number of vibration sensors reading for persistance check.

Definition at line 107 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop().

7.1.2.22 #define SERVER_OFF 2

Server powered off.

Definition at line 98 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop(), setup(), and showServerStartingStopping().

7.1.2.23 #define SERVER_ON 1

Server powered on.

Definition at line 97 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by initUptime(), loop(), and showServerStartingStopping().

7.1.2.24 #define SERVER_POWER_BTN 8

Power control simulated button pin.

Warning

This value is hardwired and should not be changed!

Definition at line 88 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by execServerPowerOff(), execServerPowerOn(), and setup().

7.1.2.25 #define SERVER_POWER_TIME 10000

msec for server going up (power On and Reset)

Definition at line 100 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop().

7.1.2.26 #define SERVER_POWEROFF_TIME 5000

msec for server goind down

Definition at line 101 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by execServerPowerOff().

7.1.2.27 #define SERVER_RESET 3

Server restarting after reset.

Definition at line 99 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop(), and showServerStartingStopping().

7.1.2.28 #define SERVER_RESET_BTN 9

Reset control simulated button pin.

Warning

This value is hardwired and should not be changed!

Definition at line 91 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by execServerReset(), and setup().

7.1.2.29 #define SERVER_RESET_TIME 5000

msec for server goind down

Definition at line 102 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop().

7.1.2.30 #define TEMP_MAX 60

Maximum temperature before overheating error If the internal temperature reach this level the board enter in a overheating risk. Leaving the system working for long time at high temperatures may produce serious damage to the components.

Definition at line 60 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by checkHealthStatus().

7.1.2.31 #define TEMP_MIN 30

Minimum temperature to start fan The temperature calculation is based on a case of about 15x15x15 cm internal size.

Note

> Take in account that the internal case temperature is expected to be lower than the controlled board temperature. Setting this limit to a too high value may seriously damage the circuits.

Definition at line 55 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by checkHealthStatus().

7.1.2.32 #define UPDATE_FREQ 10

Loop update frequency This value is a timing delay at the end of every cycle in the loop() function.

Definition at line 64 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop().

7.1.2.33 #define VIBRATION_SENSOR 0

Vibration sensor pin.

Warning

> This value is hardwired and should not be changed!

Definition at line 85 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop(), and setup().

7.1.3 Function Documentation

7.1.3.1 void checkHealthStatus ( )

Check the health status of the system and update the display.

Definition at line 349 of file CircuitHealthStatus_ControllerBoard.ino.

References BUTTON_PRESS_NONE, fanSpeed, FANSPEED_MAX, FANSPEED_MIN, fanSpeedPerc, MIN_FANSPEED_PERC, now(), prevFanSpeedPerc, prevTemp, PWM_FAN, readTemp(), showAction(), showFan(), showTemp(), temp, TEMP_MAX, TEMP_MIN, and updateTime().

Referenced by loop().

```
349                         {
350     temp = readTemp();     // get the temperature
351 
352     if(temp < TEMP_MIN) {   // if temp is lower than minimum temp
353         fanSpeed = 0;      // fan is not spinning
354         analogWrite(PWM_FAN, fanSpeed);  // spin the fan at the fanSpeed speed
355     }
356 
357     if((temp >= TEMP_MIN) && (temp <= TEMP_MAX)) {  // if temperature is higher
       than minimum temp
358         fanSpeed = map(temp, TEMP_MIN, TEMP_MAX,
      FANSPEED_MIN, FANSPEED_MAX); // the actual speed of fan
359         analogWrite(PWM_FAN, fanSpeed);  // spin the fan at the fanSpeed speed
360     }
361 
362    // Calculate the fan speed percentage
363    // as base 100 relation with the current temperature
364    fanSpeedPerc = map(temp, TEMP_MIN, TEMP_MAX, 10, 100);
365    // If fanspeed is less than 10% the shown value is forced to 0 as the
366    // applied PWM frequency is not sufficient to physically start the fan motor
367    if(fanSpeedPerc < MIN_FANSPEED_PERC)
368      fanSpeedPerc = 0;
369    // Only when changes the display value is updated
370    if(prevFanSpeedPerc != fanSpeedPerc) {
371      showFan(fanSpeedPerc);
372      prevFanSpeedPerc = fanSpeedPerc;
373    }
374    if(prevTemp != temp) {
375      showTemp(temp);
376      prevTemp = temp;
377    }
378 
379    showAction(BUTTON_PRESS_NONE);
380 
381    now((time_t)millis());
382    updateTime();
383 }
```

7.1.3.2 int checkPushReleaseButton (int *btn*)

Avoid the user keeping the button pressed.

Todo Optimize ths method with a more consistent series of samples.

Parameters

btn	The digital pin corresponding to the button

Definition at line 340 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop().

```
340                                        {
341   int pressed = digitalRead(btn);
342 
343   return pressed;
344 }
```

7.1.3.3 void clean ( )

Clean the display.

A delay is added after the hardware clear() call to give the display the time to complete the operation.

Definition at line 648 of file CircuitHealthStatus_ControllerBoard.ino.

References lcd(), and LCDCLEAR_DELAY.

```
648             {
649     lcd.clear();
650     delay(LCDCLEAR_DELAY);
651 }
```

7.1.3.4 void error (String *m*, int *x*, int *y*)

Display an error message at the specified cursor coordinates.

The error message is shown for a LCDERROR_DELAY milliseconds. After the timeout expires the screen is not cleared so the next steps should be managed by the program flow. It is expected that error messages are shown in a calling code that manages the error conditions.

Parameters

m	the message string
x	the cursor column zero based
y	the row number zero based

Definition at line 610 of file CircuitHealthStatus_ControllerBoard.ino.

References LCDERROR_DELAY, and message().

```
610                                  {
611     message(m, x, y);
612     delay(LCDERROR_DELAY);
613 }
```

7.1.3.5 void error (String *m*)

Display an error message at the cursor position.

The error message is shown for a LCDERROR_DELAY milliseconds. After the timeout expires the screen is not cleared so the next steps should be managed by the program flow. It is expected that error messages are shown in a calling code that manages the error conditions.

Parameters

m	the string message

Definition at line 625 of file CircuitHealthStatus_ControllerBoard.ino.

References LCDERROR_DELAY, and message().

```
625                       {
626     message(m);
627     delay(LCDERROR_DELAY);
628 }
```

7.1.3.6 void execServerPowerOff ( )

Send a reset signal sequence to the server.

Definition at line 326 of file CircuitHealthStatus_ControllerBoard.ino.

References POWEROFF_CYCLE_DURATION, SERVER_POWER_BTN, and SERVER_POWEROFF_TIME.

Referenced by loop().

```
326                           {
327   digitalWrite(SERVER_POWER_BTN, HIGH);
328   delay(POWEROFF_CYCLE_DURATION);
329   digitalWrite(SERVER_POWER_BTN, LOW);
330   delay(SERVER_POWEROFF_TIME);
331 }
```

7.1.3.7 void execServerPowerOn ( )

Send a power on signal sequence to the server.

Definition at line 319 of file CircuitHealthStatus_ControllerBoard.ino.

References POWERON_CYCLE_DURATION, and SERVER_POWER_BTN.

Referenced by loop().

```
319                          {
320   digitalWrite(SERVER_POWER_BTN, HIGH);
321   delay(POWERON_CYCLE_DURATION);
322   digitalWrite(SERVER_POWER_BTN, LOW);
323 }
```

7.1.3.8 void execServerReset ( )

Send a reset signal sequence to the server.

Definition at line 312 of file CircuitHealthStatus_ControllerBoard.ino.

References RESET_CYCLE_DURATION, and SERVER_RESET_BTN.

Referenced by loop().

```
312                        {
313   digitalWrite(SERVER_RESET_BTN, HIGH);
314   delay(RESET_CYCLE_DURATION);
315   digitalWrite(SERVER_RESET_BTN, LOW);
316 }
```

7.1.3.9 void initFanTemp ( )

Initialize the temperature and fan fixed text.

prevFanSpeedPerc is used to reduce the number of display updates. Initializing the variable to -90 (that never will occour in the normal conditions) the value is forced for a first update when the program start else the 0% value (fan stopped) is shown only after the fan has started at least one time.

prevTemp is used to reduce the number of display updates. Initializing the variable to an almost impossible value the startup condition forces a first update else the temperature is never shown until it does not changes at least one time.

Definition at line 535 of file CircuitHealthStatus_ControllerBoard.ino.

References _FANSPEED, _TEMPERATURE, lcd(), LCD_SECTOR1, LCD_SECTOR2, LCDTOPROW, message(), prevFanSpeedPerc, and prevTemp.

Referenced by loop(), and setup().

```
535                    {
536   lcd.clear();
537   delay(100);
538   lcd.setCursor(LCD_SECTOR1, LCDTOPROW);
539   message(_TEMPERATURE);
540   lcd.setCursor(LCD_SECTOR2, LCDTOPROW);
541   message(_FANSPEED);
542 
543   prevFanSpeedPerc = -90;
544 
545   prevTemp = -10;
546 }
```

7.1.3.10 void initUptime ( )

Initialise the Uptime string.

Definition at line 456 of file CircuitHealthStatus_ControllerBoard.ino.

References _UPTIME, _UPTIMEOFF, lcd(), LCD_SECTOR1, LCDBOTTOMROW, message(), SERVER_ON, and serverStatus.

Referenced by setup().

```
                 {
  lcd.setCursor(LCD_SECTOR1, LCDBOTTOMROW);
  if(serverStatus == SERVER_ON)
    message(_UPTIME);
  else
    message(_UPTIMEOFF);
}
```

7.1.3.11 AlphaLCD lcd (4 , 5 , 6)

LCD alphanumeric display class instance Data, latch and clock pins depends on the LCD board connection.

Warning

Don't change these settings!

Referenced by clean(), initFanTemp(), initUptime(), message(), setup(), showAction(), showFan(), showPowerOff(), showPowerOn(), showReset(), showServerStartingStopping(), showShock(), showTemp(), updateTime(), and welcome().

7.1.3.12 void loop (void)

Main loop application.

Definition at line 163 of file CircuitHealthStatus_ControllerBoard.ino.

References ALARM_TIMEOUT, BUTTON_PRESS_NONE, BUTTON_PRESS_TIMEOUT, buttonPressed, checkHealthStatus(), checkPushReleaseButton(), execServerPowerOff(), execServerPowerOn(), execServerReset(), FIRST_SHOCK_DELAY, initFanTemp(), POWER_BUTTON, POWER_ON_DELAY, PRESS_POWER_FIRST, PRESS_POWER_SECOND, PRESS_RESET_FIRST, PRESS_RESET_SECOND, RESET_BUTTON, SENSOR_READINGS, SERVER_OFF, SERVER_ON, SERVER_POWER_TIME, SERVER_RESET, SERVER_RESET_TIME, serverStatus, shockAlarmTimeout, showAction(), showPowerOff(), showPowerOn(), showReset(), showServerStartingStopping(), showShock(), startTimeSec, UPDATE_FREQ, and VIBRATION_SENSOR.

```
                {
  // ==============================
  // Check the schock risk status
  // ==============================
  // Shock alarm is checked only when the server is running
  if( (digitalRead(VIBRATION_SENSOR) == HIGH) && (serverStatus !=
  SERVER_OFF) ){
    // Shock alarm - Initialise the count
    showShock();
    // wait a few second(s) to reduce the sensor sensitivity before checking
    // for risk condition persistance. This value is calibrated experimentally
    delay(FIRST_SHOCK_DELAY);
    shockAlarmTimeout = millis();
    int numberShock; // counter of the detected vibrations
    // Exit from the alarm loop only when the alarm ends or the system shutdown process is started.
    boolean alarmSet = true;
    numberShock = 0;
    while(alarmSet) {
      // Read 100 times the sensor.
      for(int j = 0; j < SENSOR_READINGS; j++)  {
        if(digitalRead(VIBRATION_SENSOR) == HIGH)
          numberShock++;
      } // vibration counter loop
      // If alarm condition persists, update the display
      // to create a blinking effect at the end of every loop
      if(numberShock > 0)
            showShock();
      // Check for alarm timeout
      if( (millis() - shockAlarmTimeout) > ALARM_TIMEOUT)
```

```
        alarmSet = false; // just exit from the loop
    } // alarm timeout loop
    // If alarm condition persisted for too much time, the server is
    // powered off, else restore the normal conditions
    if(numberShock > 0) {
      showServerStartingStopping();
      execServerPowerOff();
      serverStatus = SERVER_OFF;
      buttonPressed = BUTTON_PRESS_NONE;
      initFanTemp();
    }
    else {
      // Restore the normal condition
      buttonPressed = BUTTON_PRESS_NONE;
      serverStatus = SERVER_ON;
      initFanTemp();
    }
  } // end vibration alarm check

  // ==============================
  // Check the state of the buttons
  // ==============================
  // Manage Reset button -----------------------------------------
  if (checkPushReleaseButton(RESET_BUTTON) == LOW) {
      showAction(RESET_BUTTON);
      // Reset button
      if( (buttonPressed == PRESS_RESET_FIRST) && (
  serverStatus == SERVER_RESET) ){
        buttonPressed = PRESS_RESET_SECOND;
      } // Second button accepted
    else {
      if(serverStatus == SERVER_ON) {
        buttonPressed = PRESS_RESET_FIRST;
      } // Server on, can reset
      else {
        buttonPressed = BUTTON_PRESS_NONE;
        initFanTemp();
      } // Server off, reset impossible
    } // First press
  } // Reset button pressed

  // Manage Power on/off button ----------------------------------
  else if(checkPushReleaseButton(POWER_BUTTON) == LOW) {
    showAction(POWER_BUTTON);
    if( (buttonPressed == PRESS_POWER_FIRST) && (
  serverStatus == SERVER_ON) ) {
        buttonPressed = PRESS_POWER_SECOND;
    } // First button already pressed with server on
    else {
      // Power on the server
        buttonPressed = PRESS_POWER_FIRST;
    }
  } // Power Button pressed

  // ==========================================
  // Process the current buttons status action
  // ==========================================
  switch(buttonPressed) {

   case BUTTON_PRESS_NONE:
    // No action request, check health status and go ahead
    checkHealthStatus();
    startTimeSec = millis(); // Initialise the timeout counter
   break;

   case PRESS_POWER_FIRST:
    if( (millis() - startTimeSec) > BUTTON_PRESS_TIMEOUT) {
      buttonPressed = BUTTON_PRESS_NONE;
      initFanTemp();
      } // First button timeout
    else {
      if(serverStatus == SERVER_OFF) {
        showPowerOn();
        execServerPowerOn();
        delay(POWER_ON_DELAY);
        showServerStartingStopping();
        delay(SERVER_POWER_TIME);  // Wait for server power on and start
        serverStatus = SERVER_ON;
        buttonPressed = BUTTON_PRESS_NONE;
        initFanTemp();
      }
      else {
        // Ask for confirmation to start poweroff sequence
        showPowerOff();
      }
    } // No timeout
   break;
```

```

    case PRESS_POWER_SECOND:
     showServerStartingStopping();
     execServerPowerOff();
     serverStatus = SERVER_OFF;
     buttonPressed = BUTTON_PRESS_NONE;
     initFanTemp();
    break;

    case PRESS_RESET_FIRST:
     if( (millis() - startTimeSec) > BUTTON_PRESS_TIMEOUT) {
       initFanTemp();
       serverStatus = SERVER_ON;
       buttonPressed = BUTTON_PRESS_NONE;
     } // First button timeout
     else {
       showReset();
       serverStatus = SERVER_RESET;
     }
    break;

    case PRESS_RESET_SECOND:
     // Reset sequence
     showServerStartingStopping();
     execServerReset();
     delay(SERVER_RESET_TIME);
     serverStatus = SERVER_ON;
     buttonPressed = BUTTON_PRESS_NONE;
     initFanTemp();
    break;
    }

     delay(UPDATE_FREQ);
}
```

7.1.3.13 void message (String *m*)

Display a string on the LCD at the cursor position.

Parameters

m	the message string

Definition at line 594 of file CircuitHealthStatus_ControllerBoard.ino.

References lcd().

Referenced by error(), initFanTemp(), initUptime(), message(), showPowerOff(), showPowerOn(), showReset(), showServerStartingStopping(), and showShock().

```
                        {
    lcd.print(m);
}
```

7.1.3.14 void message (String *m*, int *x*, int *y*)

Display a string on the LCD at the specified cursor coordinates.

Parameters

m	the string message
x	the cursor column zero based
y	the row number zero based

Definition at line 637 of file CircuitHealthStatus_ControllerBoard.ino.

References lcd(), and message().

```
                                     {
    lcd.setCursor(x, y);
    message(m);
}
```

7.1.3.15 int readTemp ( )

Read the analog value from the LM35 temperature sensor with the correction constant to convert in Celsius.

Note

Take in account that the LM35 temperature sensor is natively calibrated to provide Celsius measurements.

Definition at line 392 of file CircuitHealthStatus_ControllerBoard.ino.

References temp.

Referenced by checkHealthStatus().

```
                {
  temp = analogRead(0);
  return temp * 0.48828125;
}
```

7.1.3.16 void setup ( )

Setup method on power-on.

Definition at line 134 of file CircuitHealthStatus_ControllerBoard.ino.

References ANALOG_TEMP, BUTTON_PRESS_NONE, buttonPressed, initFanTemp(), initUptime(), lcd(), LCDCHARS, LCDROWS, now(), POWER_BUTTON, PWM_FAN, RESET_BUTTON, SERVER_OFF, SERVER_POWER_BTN, SERVER_RESET_BTN, serverStatus, testFan(), VIBRATION_SENSOR, and welcome().

```
               {
  pinMode(PWM_FAN, OUTPUT);
  pinMode(ANALOG_TEMP, INPUT);
  pinMode(RESET_BUTTON, INPUT);
  pinMode(POWER_BUTTON, INPUT);
  pinMode(VIBRATION_SENSOR, INPUT);
  pinMode(SERVER_POWER_BTN, OUTPUT);
  pinMode(SERVER_RESET_BTN, OUTPUT);

  // Initial server status
  serverStatus = SERVER_OFF;
  // Initial buttons status
  buttonPressed = BUTTON_PRESS_NONE;

  // Initializes the LCD library
  lcd.begin(LCDCHARS, LCDROWS);
  // Turn LCD On
  lcd.display();
  // Initial message
  welcome();
  initFanTemp();
  testFan();

  now((time_t)millis()); // For UpTime initialisation
  initUptime();

}
```

7.1.3.17 void showAction (int *btn*)

Notification while a user action is active through a button press sequence.

Definition at line 404 of file CircuitHealthStatus_ControllerBoard.ino.

References _ACTION_ACTIVE_POWER, _ACTION_ACTIVE_RESET, _NO_ACTION, lcd(), LCDBOTTOMROW, POWER_BUTTON, and RESET_BUTTON.

Referenced by checkHealthStatus(), and loop().

```
                        {
  lcd.setCursor(0, LCDBOTTOMROW);
  if(btn == POWER_BUTTON)
    lcd << _ACTION_ACTIVE_POWER;
  else if(btn == RESET_BUTTON)
    lcd << _ACTION_ACTIVE_RESET;
  else
    lcd << _NO_ACTION;

  delay(65);  // to draw the string
}
```

7.1.3.18 void showFan (int *dFan*)

Update the display fan speed (in percentage)

Parameters

dFan	current fan speed %

Definition at line 563 of file CircuitHealthStatus_ControllerBoard.ino.

References _SPACING, lcd(), LCD_SECTOR2, LCD_SPEED_VAL_OFFSET, and LCDTOPROW.

Referenced by checkHealthStatus().

```
                      {
  int outFan;

  if(dFan < 0)
    outFan = 0;
  else
    outFan = dFan;

  lcd.setCursor(LCD_SECTOR2 + LCD_SPEED_VAL_OFFSET,
  LCDTOPROW);
  lcd << _SPACING << _SPACING << _SPACING;
  lcd.setCursor(LCD_SECTOR2 + LCD_SPEED_VAL_OFFSET,
  LCDTOPROW);
  lcd << outFan << "%";
}
```

7.1.3.19 void showPowerOff ( )

Show the powerOff strings.

Definition at line 495 of file CircuitHealthStatus_ControllerBoard.ino.

References _POWEROFF1, _POWEROFF2, lcd(), LCD_SECTOR1, LCD_SECTOR3, LCDBOTTOMROW, LCDTOPROW, and message().

Referenced by loop().

```
                    {
  lcd.clear();
  delay(50);
  lcd.setCursor(LCD_SECTOR1, LCDTOPROW);
  message(_POWEROFF1);
  lcd.setCursor(LCD_SECTOR3, LCDBOTTOMROW);
  message(_POWEROFF2);
}
```

7.1.3.20 void showPowerOn ( )

Show the powerOn strings.

Definition at line 485 of file CircuitHealthStatus_ControllerBoard.ino.

References _POWERON1, _POWERON2, lcd(), LCD_SECTOR1, LCD_SECTOR3, LCDBOTTOMROW, LCDTOPROW, and message().

Referenced by loop().

```
                    {
  lcd.clear();
  delay(50);
  lcd.setCursor(LCD_SECTOR1, LCDTOPROW);
  message(_POWERON1);
  lcd.setCursor(LCD_SECTOR3, LCDBOTTOMROW);
  message(_POWERON2);
}
```

7.1.3.21 void showReset ( )

Show the reset strings.

Definition at line 475 of file CircuitHealthStatus_ControllerBoard.ino.

References _RESET1, _RESET2, lcd(), LCD_SECTOR1, LCD_SECTOR3, LCDBOTTOMROW, LCDTOPROW, and message().

Referenced by loop().

```
                  {
  lcd.clear();
  delay(50);
  lcd.setCursor(LCD_SECTOR1, LCDTOPROW);
  message(_RESET1);
  lcd.setCursor(LCD_SECTOR3, LCDBOTTOMROW);
  message(_RESET2);
}
```

7.1.3.22 void showServerStartingStopping ( )

Show the server starting message.

Definition at line 505 of file CircuitHealthStatus_ControllerBoard.ino.

References _EMPTY_HALF_LINE, _POWER_RESTART, _POWEROFF_RUN, _POWERON_RUN, lcd(), LCD_S-ECTOR3, LCDBOTTOMROW, message(), SERVER_OFF, SERVER_ON, SERVER_RESET, and serverStatus.

Referenced by loop().

```
                                   {
  lcd.setCursor(LCD_SECTOR3, LCDBOTTOMROW);
  lcd << _EMPTY_HALF_LINE << _EMPTY_HALF_LINE;
  lcd.setCursor(LCD_SECTOR3, LCDBOTTOMROW);

  // If server is off, it is starting else it is stopping
  // or it is restarting
  if(serverStatus == SERVER_ON)
    message(_POWEROFF_RUN);
  else if(serverStatus == SERVER_OFF)
    message(_POWERON_RUN);
  else if(serverStatus == SERVER_RESET)
    message(_POWER_RESTART);
}
```

7.1.3.23 void showShock ( )

Show the shock risk string.

Definition at line 465 of file CircuitHealthStatus_ControllerBoard.ino.

References _SHOCK1, _SHOCK2, lcd(), LCD_SECTOR1, LCD_SECTOR3, LCDBOTTOMROW, LCDTOPROW, and message().

Referenced by loop().

```
                    {
  lcd.clear();
  delay(50);
  lcd.setCursor(LCD_SECTOR1 + 1, LCDTOPROW);
  message(_SHOCK1);
  lcd.setCursor(LCD_SECTOR3 + 3, LCDBOTTOMROW);
  message(_SHOCK2);
}
```

7.1.3.24 void showTemp (int *dTemp*)

Update the display temperature.

Parameters

dTemp	current temperature

Definition at line 582 of file CircuitHealthStatus_ControllerBoard.ino.

References _SPACING, lcd(), LCD_SECTOR1, LCD_TEMP_VAL_OFFSET, and LCDTOPROW.

Referenced by checkHealthStatus().

```
                          {
  lcd.setCursor(LCD_SECTOR1 + LCD_TEMP_VAL_OFFSET,
  LCDTOPROW);
  lcd << _SPACING << _SPACING;
  lcd.setCursor(LCD_SECTOR1 + LCD_TEMP_VAL_OFFSET,
  LCDTOPROW);
  lcd << dTemp << "C";
}
```

7.1.3.25 void testFan ( )

Fan fixed text.

Definition at line 551 of file CircuitHealthStatus_ControllerBoard.ino.

References FAN_TEST_MS, FANSPEED_MAX, FANSPEED_MIN, and PWM_FAN.

Referenced by setup().

```
                 {
  // Start the fan for test at max speed
   analogWrite(PWM_FAN, FANSPEED_MAX);
   delay(FAN_TEST_MS);
   analogWrite(PWM_FAN, FANSPEED_MIN);
}
```

7.1.3.26 void updateTime ( )

Update the uptime string.

Definition at line 441 of file CircuitHealthStatus_ControllerBoard.ino.

References _SPACING, _UPTIME_DAYS, _UPTIME_SEP, day(), hour(), lcd(), LCD_SECTOR1, LCD_UPTIME_O-FFSET, LCDBOTTOMROW, minute(), and second().

Referenced by checkHealthStatus().

```
                    {
    int dd, hh, mm, ss;

    dd = day();
    hh = hour();
    mm = minute();
    ss = second();

```

```
449     lcd.setCursor(LCD_SECTOR1 + LCD_UPTIME_OFFSET,
      LCDBOTTOMROW);
450     lcd << ((dd<100)?"0":"") << ((dd<10)?"0":"") << dd << _UPTIME_DAYS <<
      _SPACING
451         << ((hh<10)?"0":"") << hh  << _UPTIME_SEP << ((mm<10)?"0":"") << mm
452         << _UPTIME_SEP << ((ss<10)?"0":"") << ss;
453 }
```

7.1.3.27 void welcome ( )

Welcome message shown at device power-on.

Definition at line 419 of file CircuitHealthStatus_ControllerBoard.ino.

References _SPACING, _VERSION, _WEB, build, lcd(), LCDBOTTOMROW, LCDCHARS, LCDMESSAGE_DELAY, LCDTOPROW, project, and version.

Referenced by setup().

```
419                {
420
421   lcd.clear();
422   lcd.setCursor(0, LCDTOPROW);
423   lcd << project();
424   lcd.setCursor(0, LCDBOTTOMROW);
425   lcd << _VERSION << _SPACING << version() << _SPACING <<
      build();
426   delay(LCDMESSAGE_DELAY);
427   lcd.clear();
428
429   lcd.setCursor(LCDCHARS, LCDTOPROW);
430   lcd.print(_WEB);
431
432   // scroll
433   for (int positionCounter = 0; positionCounter < (LCDCHARS * 2); positionCounter++) {
434     lcd.scrollDisplayLeft();
435     delay(200);
436   }
437   lcd.clear();
438 }
```

7.1.4 Variable Documentation

7.1.4.1 int buttonPressed

the current button pressed

Definition at line 118 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop(), and setup().

7.1.4.2 int fanSpeed

the current fan speed

Definition at line 121 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by checkHealthStatus().

7.1.4.3 int fanSpeedPerc

the fan speed in percentage (for visualisation)

Definition at line 122 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by checkHealthStatus().

7.1.4.4 int prevFanSpeedPerc

Last fan speed percentage.

Definition at line 123 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by checkHealthStatus(), and initFanTemp().

7.1.4.5 int prevTemp

Last temperature read.

Definition at line 124 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by checkHealthStatus(), and initFanTemp().

7.1.4.6 int serverStatus

the current status of the server

Definition at line 119 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by initUptime(), loop(), setup(), and showServerStartingStopping().

7.1.4.7 unsigned long shockAlarmTimeout

The alarm timeout counter.

Definition at line 125 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop().

7.1.4.8 unsigned long startTimeSec

Time value for button press validity telay.

Definition at line 126 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by loop().

7.1.4.9 int temp

the current temperature value

Definition at line 120 of file CircuitHealthStatus_ControllerBoard.ino.

Referenced by checkHealthStatus(), and readTemp().

7.2 /Volumes/John Doe/Development Projects/Circuit_Health_Controller/CHC/Circuit-HealthStatus_ControllerBoard/LCD.h File Reference

LCD display Manager include file.

```
#include <inttypes.h>
#include <Print.h>
#include <AlphaLCD.h>
#include <Streaming.h>
```

Classes

- class LCD

  Manages the Alphanumeric display for program output messages.

Macros

- #define LCDclockPin 4

  LCD Shift control pin - Clock signal Define this value accordingly with the Arduino board connections.
- #define LCDlatchPin 5

  LCD Shift control pin - Latch signal Define this value accordingly with the Arduino board connections.
- #define LCDdataPin 6

  LCD Shift control pin - Data signal Define this value accordingly with the Arduino board connections.
- #define LCDCHARS 16

  Display characters per line Define this value accordingly with the LCD Hardware datasheet.
- #define LCDROWS 2

  Display rows.
- #define LCDTOPROW 0

  The top row number of the LCD.
- #define LCDBOTTOMROW 1

  The bottom row number of the LCD.
- #define LCD_SECTOR1 0

  Top Left display sector column.
- #define LCD_SECTOR2 LCDCHARS / 2

  Top Right display sector column.
- #define LCD_SECTOR3 0

  Bottom Left display sector column.
- #define LCD_SECTOR4 LCDCHARS / 2

  Bottom Right display sector column.
- #define LCD_TEMP_VAL_OFFSET 5

  Temperature value offset position (right to the text) Depends on the text lenght defined in Strings.h.
- #define LCD_SPEED_VAL_OFFSET 4

  Speed perc. value offset position (right to the text) Depends on the text lenght defined in Strings.h.
- #define LCD_UPTIME_OFFSET 3

  Uptime variable text offset.
- #define LCDERROR_DELAY 5000

  Delay after showing an error.
- #define LCDMESSAGE_DELAY 5000

  Delay after showing a temporary message e.g. the welcome screen.
- #define LCDCLEAR_DELAY 50

  Delay after a clear display call to hardware has been done.

7.2.1 Detailed Description

LCD display Manager include file. Methods to manage the LCD output and display features, including some hard-coded strings like the welcome message.

Definition in file LCD.h.

7.2.2 Macro Definition Documentation

7.2.2.1 #define LCD_SECTOR1 0

Top Left display sector column.

Definition at line 35 of file LCD.h.

Referenced by initFanTemp(), initUptime(), showPowerOff(), showPowerOn(), showReset(), showShock(), showTemp(), and updateTime().

7.2.2.2 #define LCD_SECTOR2 LCDCHARS / 2

Top Right display sector column.

Definition at line 37 of file LCD.h.

Referenced by initFanTemp(), and showFan().

7.2.2.3 #define LCD_SECTOR3 0

Bottom Left display sector column.

Definition at line 39 of file LCD.h.

Referenced by showPowerOff(), showPowerOn(), showReset(), showServerStartingStopping(), and showShock().

7.2.2.4 #define LCD_SECTOR4 LCDCHARS / 2

Bottom Right display sector column.

Definition at line 41 of file LCD.h.

7.2.2.5 #define LCD_SPEED_VAL_OFFSET 4

Speed perc. value offset position (right to the text) Depends on the text lenght defined in Strings.h.

Definition at line 48 of file LCD.h.

Referenced by showFan().

7.2.2.6 #define LCD_TEMP_VAL_OFFSET 5

Temperature value offset position (right to the text) Depends on the text lenght defined in Strings.h.

Definition at line 45 of file LCD.h.

Referenced by showTemp().

7.2.2.7 #define LCD_UPTIME_OFFSET 3

Uptime variable text offset.

Definition at line 50 of file LCD.h.

Referenced by updateTime().

7.2.2.8 #define LCDBOTTOMROW 1

The bottom row number of the LCD.

Definition at line 33 of file LCD.h.

Referenced by initUptime(), showAction(), showPowerOff(), showPowerOn(), showReset(), showServerStarting-Stopping(), showShock(), updateTime(), and welcome().

7.2.2.9 #define LCDCHARS 16

Display characters per line Define this value accordingly with the LCD Hardware datasheet.

Definition at line 27 of file LCD.h.

Referenced by setup(), and welcome().

7.2.2.10 #define LCDCLEAR_DELAY 50

Delay after a clear display call to hardware has been done.

Definition at line 57 of file LCD.h.

Referenced by clean().

7.2.2.11 #define LCDclockPin 4

LCD Shift control pin - Clock signal Define this value accordingly with the Arduino board connections.

Definition at line 18 of file LCD.h.

7.2.2.12 #define LCDdataPin 6

LCD Shift control pin - Data signal Define this value accordingly with the Arduino board connections.

Definition at line 24 of file LCD.h.

7.2.2.13 #define LCDERROR_DELAY 5000

Delay after showing an error.

Definition at line 53 of file LCD.h.

Referenced by error().

7.2.2.14 #define LCDlatchPin 5

LCD Shift control pin - Latch signal Define this value accordingly with the Arduino board connections.

Definition at line 21 of file LCD.h.

7.2.2.15 #define LCDMESSAGE_DELAY 5000

Delay after showing a temporary message e.g. the welcome screen.

Definition at line 55 of file LCD.h.

Referenced by welcome().

7.2.2.16 #define LCDROWS 2

Display rows.

Definition at line 29 of file LCD.h.

Referenced by setup().

7.2.2.17 #define LCDTOPROW 0

The top row number of the LCD.

Definition at line 31 of file LCD.h.

Referenced by initFanTemp(), showFan(), showPowerOff(), showPowerOn(), showReset(), showShock(), showTemp(), and welcome().

7.3 /Volumes/John Doe/Development Projects/Circuit_Health_Controller/CHC/CircuitHealthStatus_ControllerBoard/Strings.h File Reference

LCD Display base strings.

```
#include "Version.h"
```

Macros

- #define _SPACING " "
- #define _EMPTY_HALF_LINE " "
- #define _WEB "balearicdynamics.com"
- #define _VERSION "Ver."
- #define _ACTION_ACTIVE_POWER "[]"
- #define _ACTION_ACTIVE_RESET "]["
- #define _NO_ACTION "--"
- #define _TEMPERATURE "Temp."
- #define _FANSPEED " Sp."
- #define _UPTIME "On"
- #define _UPTIMEOFF "--"
- #define _UPTIME_DAYS "d"
- #define _UPTIME_SEP ":"
- #define _RESET1 "Server Reset"
- #define _RESET2 "Press to confirm"
- #define _POWER_RESTART "Server resetting"
- #define _POWERON1 "Power on"
- #define _POWERON2 "Start server"
- #define _POWERON_RUN "Server starting"
- #define _POWEROFF1 "Server off"
- #define _POWEROFF2 "Press to confirm"
- #define _POWEROFF_RUN "System stopping"
- #define _SHOCK1 "∗∗∗ ALARM! ∗∗∗"
- #define _SHOCK2 "Shock Risk"

7.3.1 Detailed Description

LCD Display base strings. The strings used to build the board interface.

Definition in file Strings.h.

7.3.2 Macro Definition Documentation

7.3.2.1 #define _ACTION_ACTIVE_POWER "[ ]"

Definition at line 19 of file Strings.h.

Referenced by showAction().

7.3.2.2 #define _ACTION_ACTIVE_RESET "]["

Definition at line 20 of file Strings.h.

Referenced by showAction().

7.3.2.3 #define _EMPTY_HALF_LINE " "

Definition at line 14 of file Strings.h.

Referenced by showServerStartingStopping().

7.3.2.4 #define _FANSPEED " Sp."

Definition at line 24 of file Strings.h.

Referenced by initFanTemp().

7.3.2.5 #define _NO_ACTION "--"

Definition at line 21 of file Strings.h.

Referenced by showAction().

7.3.2.6 #define _POWER_RESTART "Server resetting"

Definition at line 33 of file Strings.h.

Referenced by showServerStartingStopping().

7.3.2.7 #define _POWEROFF1 "Server off"

Definition at line 39 of file Strings.h.

Referenced by showPowerOff().

7.3.2.8 #define _POWEROFF2 "Press to confirm"

Definition at line 40 of file Strings.h.

Referenced by showPowerOff().

7.3.2.9 #define _POWEROFF_RUN "System stopping"

Definition at line 41 of file Strings.h.

Referenced by showServerStartingStopping().

7.3.2.10 #define _POWERON1 "Power on"

Definition at line 35 of file Strings.h.

Referenced by showPowerOn().

7.3.2.11 #define _POWERON2 "Start server"

Definition at line 36 of file Strings.h.

Referenced by showPowerOn().

7.3.2.12 #define _POWERON_RUN "Server starting"

Definition at line 37 of file Strings.h.

Referenced by showServerStartingStopping().

7.3.2.13 #define _RESET1 "Server Reset"

Definition at line 31 of file Strings.h.

Referenced by showReset().

7.3.2.14 #define _RESET2 "Press to confirm"

Definition at line 32 of file Strings.h.

Referenced by showReset().

7.3.2.15 #define _SHOCK1 "∗∗∗ ALARM! ∗∗∗"

Definition at line 43 of file Strings.h.

Referenced by showShock().

7.3.2.16 #define _SHOCK2 "Shock Risk"

Definition at line 44 of file Strings.h.

Referenced by showShock().

7.3.2.17 #define _SPACING " "

Definition at line 13 of file Strings.h.

Referenced by showFan(), showTemp(), updateTime(), and welcome().

7.3.2.18 #define _TEMPERATURE "Temp."

Definition at line 23 of file Strings.h.

Referenced by initFanTemp().

7.3.2.19 #define _UPTIME "On"

Definition at line 26 of file Strings.h.

Referenced by initUptime().

7.3.2.20 #define _UPTIME_DAYS "d"

Definition at line 28 of file Strings.h.

Referenced by updateTime().

7.3.2.21 #define _UPTIME_SEP ":"

Definition at line 29 of file Strings.h.

Referenced by updateTime().

7.3.2.22 #define _UPTIMEOFF "--"

Definition at line 27 of file Strings.h.

Referenced by initUptime().

7.3.2.23 #define _VERSION "Ver."

Definition at line 17 of file Strings.h.

Referenced by welcome().

7.3.2.24 #define _WEB "balearicdynamics.com"

Definition at line 16 of file Strings.h.

Referenced by welcome().

7.4 /Volumes/John Doe/Development Projects/Circuit_Health_Controller/CHC/Circuit-HealthStatus_ControllerBoard/UpTime.h File Reference

Time constans, macros and functions prototypes.

```
#include <inttypes.h>
#include <sys/types.h>
```

Classes

- struct tmElements_t

    Structure defining the used millis() converted values in the proper format.

Macros

- #define SECS_PER_MIN (60UL)
- #define SECS_PER_HOUR (3600UL)

- #define SECS_PER_DAY (SECS_PER_HOUR ∗ 24UL)
- #define numberOfSeconds(_time_) (_time_ % SECS_PER_MIN)

  Macros for fast elapsed time calculation.
- #define numberOfMinutes(_time_) ((_time_ / SECS_PER_MIN) % SECS_PER_MIN)
- #define numberOfHours(_time_) ((_time_% SECS_PER_DAY) / SECS_PER_HOUR)

Typedefs

- typedef unsigned long time_t
- typedef struct tmElements_t TimeElements
- typedef struct tmElements_t ∗ tmElementsPtr_t
- typedef time_t(∗ getExternalTime)()

Enumerations

- enum tmByteFields { tmSecond, tmMinute, tmHour, tmDay }

  Enumerator to identify the time types.

Functions

- int hour ()

  the hour now
- int hour (time_t t)

  the hour for the given time
- int minute ()

  the minute now
- int minute (time_t t)

  the minute for the given time
- int second ()

  the second now
- int second (time_t t)

  the second for the given time
- int day ()

  the day now
- int day (time_t t)

  the day for the given time
- void now (unsigned long ms)
- time_t now ()
- void adjustTime (long adjustment)
- void breakTime (time_t time, tmElements_t &tm)

  break time_t into elements
- time_t makeTime (tmElements_t &tm)

  convert time elements into time_t

7.4.1 Detailed Description

Time constans, macros and functions prototypes.

Definition in file UpTime.h.

7.4.2 Macro Definition Documentation

7.4.2.1 #define numberOfHours(*_time_*) ((_time_% SECS_PER_DAY) / SECS_PER_HOUR)

Definition at line 41 of file UpTime.h.

7.4.2.2 #define numberOfMinutes(*_time_*) ((_time_ / SECS_PER_MIN) % SECS_PER_MIN)

Definition at line 40 of file UpTime.h.

7.4.2.3 #define numberOfSeconds(*_time_*) (_time_ % SECS_PER_MIN)

Macros for fast elapsed time calculation.

Definition at line 39 of file UpTime.h.

7.4.2.4 #define SECS_PER_DAY (SECS_PER_HOUR * 24UL)

Definition at line 36 of file UpTime.h.

7.4.2.5 #define SECS_PER_HOUR (3600UL)

Definition at line 35 of file UpTime.h.

7.4.2.6 #define SECS_PER_MIN (60UL)

Definition at line 34 of file UpTime.h.

7.4.3 Typedef Documentation

7.4.3.1 typedef time_t(* getExternalTime)()

Definition at line 32 of file UpTime.h.

7.4.3.2 typedef unsigned long time_t

Definition at line 15 of file UpTime.h.

7.4.3.3 typedef struct tmElements_t TimeElements

7.4.3.4 typedef struct tmElements_t * tmElementsPtr_t

7.4.4 Enumeration Type Documentation

7.4.4.1 enum tmByteFields

Enumerator to identify the time types.

Enumerator

tmSecond

tmMinute

tmHour

tmDay

Definition at line 19 of file UpTime.h.

```
               {
    tmSecond, tmMinute, tmHour, tmDay
} tmByteFields;
```

7.4.5 Function Documentation

7.4.5.1 void adjustTime (long *adjustment*)

7.4.5.2 void breakTime (time_t *time,* tmElements_t & *tm*)

break time_t into elements

7.4.5.3 int day ( )

the day now

Referenced by updateTime().

7.4.5.4 int day (time_t *t*)

the day for the given time

7.4.5.5 int hour ( )

the hour now

Referenced by updateTime().

7.4.5.6 int hour (time_t *t*)

the hour for the given time

7.4.5.7 time_t makeTime (tmElements_t & *tm*)

convert time elements into time_t

7.4.5.8 int minute ( )

the minute now

Referenced by updateTime().

7.4.5.9 int minute (time_t *t*)

the minute for the given time

7.4.5.10 void now (unsigned long *ms*)

Referenced by checkHealthStatus(), and setup().

7.4.5.11 time_t now ( )

7.4.5.12 int second ( )

the second now

Referenced by updateTime().

7.4.5.13 int second (time_t *t*)

the second for the given time

7.5 /Volumes/John Doe/Development Projects/Circuit_Health_Controller/CHC/Circuit-HealthStatus_ControllerBoard/Version.h File Reference

Version and Build Number Helper Class.

Macros

- #define build() "1.2.4"

  Incremental build number.
- #define version() "1.4"

  Firmware version.
- #define project() "Circuit Control"

  Project name.

7.5.1 Detailed Description

Version and Build Number Helper Class. This helper macros exposes the static methods to get the firmware version and the build number. Use the build() and version() metho5ds anywhere in the program including this file

Definition in file Version.h.

7.5.2 Macro Definition Documentation

7.5.2.1 #define build( ) "1.2.4"

Incremental build number.

Definition at line 12 of file Version.h.

Referenced by welcome().

7.5.2.2 #define project( ) "Circuit Control"

Project name.

Definition at line 16 of file Version.h.

Referenced by welcome().

7.5.2.3 #define version( ) "1.4"

Firmware version.

Definition at line 14 of file Version.h.

Referenced by welcome().

Index