Arduino

1.0

Generated by Doxygen 1.8.14

Contents

1 Class Index **1**

1.1 Class List 1

2 File Index **3**

2.1 File List 3

3 Class Documentation **5**

3.1 CpuLoad Class Reference 5

3.1.1 Detailed Description 5

3.1.2 Constructor & Destructor Documentation 6

3.1.2.1 CpuLoad() 6

3.1.3 Member Function Documentation 6

3.1.3.1 ComputeCPULoad() 6

3.1.3.2 getAverageCPULoad() 7

3.1.3.3 getCurrrentCPULoad() 7

3.1.3.4 getMaxCPULoad() 8

3.1.4 Member Data Documentation 8

3.1.4.1 avg_load 8

3.1.4.2 current_load 8

3.1.4.3 last_sum_value 9

3.1.4.4 max_load 9

3.1.4.5 sample_cnt 9

3.1.4.6 sample_idx 9

3.1.4.7 sample_mem 9

3.2 DebugInterface Class Reference 10
3.2.1 Detailed Description 10
3.2.2 Constructor & Destructor Documentation 11
3.2.2.1 DebugInterface() 11
3.2.3 Member Function Documentation 11
3.2.3.1 nextLine() 11
3.2.3.2 read() 12
3.2.3.3 sendBool() 12
3.2.3.4 sendChar() 13
3.2.3.5 sendInteger() 14
3.2.3.6 sendString() [1/2] 14
3.2.3.7 sendString() [2/2] 15
3.2.4 Member Data Documentation 16
3.2.4.1 usart_drv_ptr 16
3.3 DebugManagement Class Reference 17
3.3.1 Detailed Description 18
3.3.2 Constructor & Destructor Documentation 18
3.3.2.1 DebugManagement() 18
3.3.3 Member Function Documentation 18
3.3.3.1 DebugModeManagement() 19
3.3.3.2 DisplayCPULoad_task() 20
3.3.3.3 DisplaySensors_task() 20
3.3.3.4 getIftPtr() 21
3.3.4 Member Data Documentation 22
3.3.4.1 debug_ift_ptr 22
3.3.4.2 debug_state 22
3.4 dht22 Class Reference 22
3.4.1 Detailed Description 23
3.4.2 Constructor & Destructor Documentation 23
3.4.2.1 dht22() 23

3.4.3 Member Function Documentation . . . 23
3.4.3.1 initializeCommunication() . . . 24
3.4.3.2 read() . . . 24
3.5 dio Class Reference . . . 25
3.5.1 Detailed Description . . . 26
3.5.2 Constructor & Destructor Documentation . . . 26
3.5.2.1 dio() . . . 26
3.5.3 Member Function Documentation . . . 27
3.5.3.1 dio_changePortPinCnf() . . . 27
3.5.3.2 dio_getPort() . . . 28
3.5.3.3 dio_getPort_fast() . . . 28
3.5.3.4 dio_invertPort() . . . 29
3.5.3.5 dio_memorizePINaddress() . . . 30
3.5.3.6 dio_setPort() . . . 31
3.5.3.7 getDDRxAddress() . . . 31
3.5.3.8 getPINxAddress() . . . 32
3.5.3.9 getPORTxAddress() . . . 33
3.5.3.10 ports_init() . . . 33
3.5.4 Member Data Documentation . . . 34
3.5.4.1 PINx_addr_mem . . . 34
3.5.4.2 PINx_idx_mem . . . 34
3.6 DisplayInterface Class Reference . . . 34
3.6.1 Detailed Description . . . 35
3.6.2 Constructor & Destructor Documentation . . . 36
3.6.2.1 DisplayInterface() . . . 36
3.6.3 Member Function Documentation . . . 36
3.6.3.1 ClearFullScreen() . . . 36
3.6.3.2 ClearLine() . . . 37
3.6.3.3 ClearStringInDataStruct() . . . 38
3.6.3.4 DisplayFullLine() . . . 39

3.6.3.5 FindFirstCharAddr() 40
3.6.3.6 getDisplayDataPtr() 40
3.6.3.7 IsLineEmpty() 41
3.6.3.8 RefreshLine() 41
3.6.3.9 setLineAlignment() 42
3.6.3.10 setLineAlignmentAndRefresh() 43
3.6.3.11 shiftLine_task() 43
3.6.3.12 updateLineAndRefresh() 44
3.6.4 Member Data Documentation 45
3.6.4.1 display_data 45
3.6.4.2 dummy 45
3.6.4.3 isShiftInProgress 45
3.6.4.4 p_lcd 46
3.7 DisplayManagement Class Reference 46
3.7.1 Detailed Description 47
3.7.2 Constructor & Destructor Documentation 47
3.7.2.1 DisplayManagement() 47
3.7.3 Member Function Documentation 48
3.7.3.1 DisplaySensorData_Task() 48
3.7.3.2 GetIftPointer() 49
3.7.3.3 GetTempSensorPtr() 50
3.7.3.4 RemoveWelcomeMessage_Task() 50
3.7.4 Member Data Documentation 51
3.7.4.1 p_display_ift 51
3.7.4.2 p_tempSensor 51
3.8 I2C Class Reference 51
3.8.1 Detailed Description 52
3.8.2 Constructor & Destructor Documentation 52
3.8.2.1 I2C() 52
3.8.3 Member Function Documentation 53

3.8.3.1 initializeBus() . . . 53
3.8.3.2 setBitRate() . . . 53
3.8.3.3 setTxAddress() . . . 54
3.8.3.4 writeByte() . . . 54
3.8.4 Member Data Documentation . . . 55
3.8.4.1 bitrate . . . 55
3.8.4.2 tx_address . . . 55
3.9 keepAliveLed Class Reference . . . 55
3.9.1 Detailed Description . . . 56
3.9.2 Constructor & Destructor Documentation . . . 56
3.9.2.1 keepAliveLed() . . . 56
3.9.3 Member Function Documentation . . . 56
3.9.3.1 blinkLed_task() . . . 57
3.10 LCD Class Reference . . . 57
3.10.1 Detailed Description . . . 59
3.10.2 Constructor & Destructor Documentation . . . 59
3.10.2.1 LCD() . . . 59
3.10.3 Member Function Documentation . . . 60
3.10.3.1 command() . . . 60
3.10.3.2 ConfigureBacklight() . . . 61
3.10.3.3 ConfigureCursorBlink() . . . 62
3.10.3.4 ConfigureCursorOnOff() . . . 62
3.10.3.5 ConfigureDisplayOnOff() . . . 63
3.10.3.6 ConfigureEntryModeDir() . . . 63
3.10.3.7 ConfigureEntryModeShift() . . . 64
3.10.3.8 ConfigureFontType() . . . 65
3.10.3.9 ConfigureI2CAddr() . . . 66
3.10.3.10 ConfigureLineNumber() . . . 66
3.10.3.11 GetDDRAMAddress() . . . 67
3.10.3.12 GetLineNumberCnf() . . . 67

3.10.3.13 InitializeScreen() . . . 68
3.10.3.14 SetDDRAMAddress() . . . 68
3.10.3.15 write() . . . 69
3.10.3.16 write4bits() . . . 70
3.10.3.17 WriteInRam() . . . 71
3.10.4 Member Data Documentation . . . 71
3.10.4.1 backlight_enable . . . 72
3.10.4.2 cnfCursorBlink . . . 72
3.10.4.3 cnfCursorOnOff . . . 72
3.10.4.4 cnfDisplayOnOff . . . 72
3.10.4.5 cnfEntryModeDir . . . 72
3.10.4.6 cnfEntryModeShift . . . 73
3.10.4.7 cnfFontType . . . 73
3.10.4.8 cnfI2C_addr . . . 73
3.10.4.9 cnfLineNumber . . . 73
3.10.4.10 ddram_addr . . . 73
3.10.4.11 i2c_drv_ptr . . . 74
3.11 LinkedList Class Reference . . . 74
3.11.1 Detailed Description . . . 75
3.11.2 Member Typedef Documentation . . . 75
3.11.2.1 T_LL_element . . . 76
3.11.3 Constructor & Destructor Documentation . . . 76
3.11.3.1 LinkedList() . . . 76
3.11.3.2 ∼LinkedList() . . . 76
3.11.4 Member Function Documentation . . . 77
3.11.4.1 AttachNewElement() . . . 77
3.11.4.2 FindElement() . . . 77
3.11.4.3 getCurrentElement() . . . 78
3.11.4.4 IsLLEmpty() . . . 79
3.11.4.5 MoveToNextElement() . . . 79

3.11.4.6 RemoveElement() . . . 80
3.11.4.7 ResetElementPtr() . . . 80
3.11.5 Member Data Documentation . . . 81
3.11.5.1 curElement_ptr . . . 81
3.11.5.2 firstElement . . . 81
3.12 scheduler Class Reference . . . 81
3.12.1 Detailed Description . . . 82
3.12.2 Member Typedef Documentation . . . 83
3.12.2.1 Task_t . . . 83
3.12.3 Constructor & Destructor Documentation . . . 83
3.12.3.1 scheduler() . . . 83
3.12.4 Member Function Documentation . . . 83
3.12.4.1 addPeriodicTask() . . . 83
3.12.4.2 getPitNumber() . . . 84
3.12.4.3 launchPeriodicTasks() . . . 85
3.12.4.4 LLElementCompare() . . . 85
3.12.4.5 removePeriodicTask() . . . 86
3.12.4.6 startScheduling() . . . 87
3.12.4.7 updateTaskPeriod() . . . 88
3.12.5 Member Data Documentation . . . 89
3.12.5.1 pit_number . . . 89
3.12.5.2 TasksLL_ptr . . . 89
3.13 String Class Reference . . . 90
3.13.1 Detailed Description . . . 90
3.13.2 Constructor & Destructor Documentation . . . 91
3.13.2.1 String() [1/2] . . . 91
3.13.2.2 String() [2/2] . . . 91
3.13.2.3 ~String() . . . 92
3.13.3 Member Function Documentation . . . 92
3.13.3.1 appendBool() . . . 92

3.13.3.2 appendChar() 93
3.13.3.3 appendInteger() 94
3.13.3.4 appendString() 95
3.13.3.5 Clear() 96
3.13.3.6 ComputeStringSize() 96
3.13.3.7 getSize() 97
3.13.3.8 getString() 98
3.13.4 Member Data Documentation 98
3.13.4.1 size 98
3.13.4.2 string 98
3.14 T_ASW_cnf_struct Struct Reference 99
3.14.1 Detailed Description 99
3.14.2 Member Data Documentation 99
3.14.2.1 p_DebugInterface 99
3.14.2.2 p_DebugManagement 100
3.14.2.3 p_DisplayInterface 100
3.14.2.4 p_DisplayManagement 100
3.14.2.5 p_keepAliveLed 100
3.14.2.6 p_TempSensor 100
3.15 T_BSW_cnf_struct Struct Reference 101
3.15.1 Detailed Description 101
3.15.2 Member Data Documentation 101
3.15.2.1 p_cpuload 101
3.15.2.2 p_dht22 102
3.15.2.3 p_dio 102
3.15.2.4 p_i2c 102
3.15.2.5 p_lcd 102
3.15.2.6 p_timer 102
3.15.2.7 p_usart 103
3.16 T_display_data Struct Reference 103

3.16.1 Detailed Description 104
3.16.2 Member Data Documentation 104
3.16.2.1 alignment 104
3.16.2.2 display_str 104
3.16.2.3 isEmpty 104
3.16.2.4 mode 104
3.16.2.5 shift_data 105
3.17 T_Display_shift_data Struct Reference 105
3.17.1 Detailed Description 105
3.17.2 Member Data Documentation 106
3.17.2.1 str_cur_ptr 106
3.17.2.2 str_ptr 106
3.17.2.3 temporization 106
3.18 T_LCD_conf_struct Struct Reference 106
3.18.1 Detailed Description 107
3.18.2 Member Data Documentation 107
3.18.2.1 backlight_en 107
3.18.2.2 cursor_en 107
3.18.2.3 cursorBlink_en 107
3.18.2.4 display_en 108
3.18.2.5 entryModeDir 108
3.18.2.6 entryModeShift 108
3.18.2.7 fontType_cnf 108
3.18.2.8 i2c_addr 108
3.18.2.9 i2c_bitrate 109
3.18.2.10 lineNumber_cnf 109
3.19 LinkedList::T_LL_element Struct Reference 109
3.19.1 Detailed Description 109
3.19.2 Member Data Documentation 110
3.19.2.1 data_ptr 110

3.19.2.2 nextElement . . . 110
3.20 scheduler::Task_t Struct Reference . . . 110
3.20.1 Detailed Description . . . 110
3.20.2 Member Data Documentation . . . 110
3.20.2.1 period . . . 111
3.20.2.2 TaskPtr . . . 111
3.21 TempSensor Class Reference . . . 111
3.21.1 Detailed Description . . . 112
3.21.2 Constructor & Destructor Documentation . . . 112
3.21.2.1 TempSensor() . . . 112
3.21.3 Member Function Documentation . . . 113
3.21.3.1 GetHumDecimal() . . . 113
3.21.3.2 getHumidity() . . . 113
3.21.3.3 GetHumInteger() . . . 114
3.21.3.4 getHumPtr() . . . 114
3.21.3.5 getTaskPeriod() . . . 115
3.21.3.6 getTemp() . . . 115
3.21.3.7 GetTempDecimal() . . . 116
3.21.3.8 GetTempInteger() . . . 116
3.21.3.9 getTempPtr() . . . 117
3.21.3.10 GetValidity() . . . 117
3.21.3.11 readTempSensor_task() . . . 118
3.21.3.12 setValidity() . . . 118
3.21.3.13 updateLastValidValues() . . . 119
3.21.3.14 updateTaskPeriod() . . . 119
3.21.4 Member Data Documentation . . . 120
3.21.4.1 read_humidity . . . 120
3.21.4.2 read_temperature . . . 121
3.21.4.3 task_period . . . 121
3.21.4.4 valid_hum . . . 121

3.21.4.5 valid_pit 121
3.21.4.6 valid_temp 121
3.21.4.7 validity 122
3.21.4.8 validity_last_read 122
3.22 timer Class Reference 122
3.22.1 Detailed Description 123
3.22.2 Constructor & Destructor Documentation 123
3.22.2.1 timer() 123
3.22.3 Member Function Documentation 123
3.22.3.1 configureTimer1() 123
3.22.3.2 getTimer1Value() 124
3.22.3.3 startTimer1() 125
3.22.3.4 stopTimer1() 125
3.22.4 Member Data Documentation 125
3.22.4.1 prescaler 125
3.23 usart Class Reference 126
3.23.1 Detailed Description 126
3.23.2 Constructor & Destructor Documentation 126
3.23.2.1 usart() 126
3.23.3 Member Function Documentation 127
3.23.3.1 setBaudRate() 127
3.23.3.2 usart_init() 128
3.23.3.3 usart_read() 128
3.23.3.4 usart_sendByte() 128
3.23.3.5 usart_sendString() 129
3.23.3.6 usart_transmit() 130
3.23.4 Member Data Documentation 131
3.23.4.1 BaudRate 131

4 File Documentation 133
4.1 asw.cpp File Reference . . . 133
4.1.1 Detailed Description . . . 134
4.1.2 Function Documentation . . . 134
4.1.2.1 asw_init() . . . 134
4.1.3 Variable Documentation . . . 134
4.1.3.1 ASW_cnf_struct . . . 135
4.2 asw.h File Reference . . . 135
4.2.1 Detailed Description . . . 135
4.2.2 Function Documentation . . . 136
4.2.2.1 asw_init() . . . 136
4.2.3 Variable Documentation . . . 136
4.2.3.1 ASW_cnf_struct . . . 136
4.3 bsw.cpp File Reference . . . 137
4.3.1 Detailed Description . . . 137
4.3.2 Function Documentation . . . 137
4.3.2.1 bsw_init() . . . 138
4.3.3 Variable Documentation . . . 138
4.3.3.1 BSW_cnf_struct . . . 138
4.4 bsw.h File Reference . . . 138
4.4.1 Detailed Description . . . 139
4.4.2 Function Documentation . . . 139
4.4.2.1 bsw_init() . . . 139
4.4.3 Variable Documentation . . . 140
4.4.3.1 BSW_cnf_struct . . . 140
4.5 CpuLoad.cpp File Reference . . . 140
4.5.1 Detailed Description . . . 140
4.6 CpuLoad.h File Reference . . . 141
4.6.1 Detailed Description . . . 141
4.6.2 Macro Definition Documentation . . . 141

4.6.2.1 NB_OF_SAMPLES . . . 141
4.7 DebugInterface.cpp File Reference . . . 142
4.8 DebugInterface.h File Reference . . . 142
4.8.1 Macro Definition Documentation . . . 142
4.8.1.1 USART_BAUDRATE . . . 143
4.9 DebugManagement.cpp File Reference . . . 143
4.9.1 Detailed Description . . . 143
4.9.2 Variable Documentation . . . 144
4.9.2.1 str_debug_main_menu . . . 144
4.10 DebugManagement.h File Reference . . . 144
4.10.1 Detailed Description . . . 145
4.10.2 Macro Definition Documentation . . . 145
4.10.2.1 PERIOD_MS_TASK_DISPLAY_CPU_LOAD . . . 145
4.10.2.2 PERIOD_MS_TASK_DISPLAY_SENSORS . . . 145
4.10.3 Enumeration Type Documentation . . . 145
4.10.3.1 debug_state_t . . . 145
4.11 dht22.cpp File Reference . . . 146
4.11.1 Detailed Description . . . 146
4.11.2 Macro Definition Documentation . . . 146
4.11.2.1 MAX_WAIT_TIME_US . . . 147
4.12 dht22.h File Reference . . . 147
4.12.1 Detailed Description . . . 147
4.12.2 Macro Definition Documentation . . . 147
4.12.2.1 DHT22_PORT . . . 148
4.13 dio.cpp File Reference . . . 148
4.13.1 Detailed Description . . . 148
4.14 dio.h File Reference . . . 148
4.14.1 Detailed Description . . . 149
4.14.2 Macro Definition Documentation . . . 149
4.14.2.1 DECODE_PIN . . . 149

4.14.2.2 DECODE_PORT 150
4.14.2.3 ENCODE_PORT 150
4.14.2.4 PORT_A 150
4.14.2.5 PORT_B 150
4.14.2.6 PORT_C 150
4.14.2.7 PORT_CNF_IN 151
4.14.2.8 PORT_CNF_OUT 151
4.14.2.9 PORT_D 151
4.15 dio_port_cnf.h File Reference 151
4.15.1 Detailed Description 151
4.15.2 Macro Definition Documentation 152
4.15.2.1 PORTB_CNF_DDRB 152
4.15.2.2 PORTB_CNF_PORTB 152
4.16 dio_reg_atm2560.h File Reference 152
4.16.1 Macro Definition Documentation 153
4.16.1.1 DDRA_PTR 153
4.16.1.2 DDRB_PTR 153
4.16.1.3 DDRC_PTR 153
4.16.1.4 DDRD_PTR 153
4.16.1.5 PINA_PTR 153
4.16.1.6 PINB_PTR 154
4.16.1.7 PINC_PTR 154
4.16.1.8 PIND_PTR 154
4.16.1.9 PORTA_PTR 154
4.16.1.10 PORTB_PTR 154
4.16.1.11 PORTC_PTR 155
4.16.1.12 PORTD_PTR 155
4.17 DisplayInterface.cpp File Reference 155
4.17.1 Detailed Description 156
4.18 DisplayInterface.h File Reference 156

4.18.1 Detailed Description . . . 157
4.18.2 Macro Definition Documentation . . . 157
4.18.2.1 DISPLAY_LINE_SHIFT_PERIOD_MS . . . 157
4.18.2.2 DISPLAY_LINE_SHIFT_TEMPO_TIME . . . 157
4.18.3 Enumeration Type Documentation . . . 157
4.18.3.1 T_DisplayInterface_LineAlignment . . . 157
4.18.3.2 T_DisplayInterface_LineDisplayMode . . . 158
4.19 DisplayManagement.cpp File Reference . . . 158
4.19.1 Detailed Description . . . 159
4.20 DisplayManagement.h File Reference . . . 159
4.20.1 Detailed Description . . . 160
4.20.2 Macro Definition Documentation . . . 160
4.20.2.1 DISPLAY_MGT_I2C_BITRATE . . . 160
4.20.2.2 DISPLAY_MGT_LCD_I2C_ADDR . . . 160
4.20.2.3 DISPLAY_MGT_LINE_HUM . . . 161
4.20.2.4 DISPLAY_MGT_LINE_TEMP . . . 161
4.20.2.5 DISPLAY_MGT_PERIOD_TASK_SENSOR . . . 161
4.20.2.6 DISPLAY_MGT_PERIOD_WELCOME_MSG_REMOVAL . . . 161
4.20.3 Variable Documentation . . . 161
4.20.3.1 humidityDisplayString . . . 161
4.20.3.2 LCD_init_cnf . . . 162
4.20.3.3 tempDisplayString . . . 162
4.20.3.4 welcomeMessageString . . . 162
4.21 I2C.cpp File Reference . . . 162
4.21.1 Detailed Description . . . 163
4.22 I2C.h File Reference . . . 163
4.22.1 Detailed Description . . . 163
4.22.2 Macro Definition Documentation . . . 164
4.22.2.1 DATA_ACK . . . 164
4.22.2.2 SLA_ACK . . . 164

4.22.2.3 START 164
4.23 keepAliveLed.cpp File Reference 164
4.23.1 Detailed Description 165
4.24 keepAliveLed.h File Reference 165
4.24.1 Detailed Description 165
4.24.2 Macro Definition Documentation 166
4.24.2.1 LED_PORT 166
4.24.2.2 PERIOD_MS_TASK_LED 166
4.25 LCD.cpp File Reference 166
4.25.1 Detailed Description 167
4.26 LCD.h File Reference 167
4.26.1 Detailed Description 168
4.26.2 Macro Definition Documentation 169
4.26.2.1 BACKLIGHT_PIN 169
4.26.2.2 EN_PIN 169
4.26.2.3 LCD_CNF_BACKLIGHT_OFF 169
4.26.2.4 LCD_CNF_BACKLIGHT_ON 169
4.26.2.5 LCD_CNF_CURSOR_BLINK_OFF 169
4.26.2.6 LCD_CNF_CURSOR_BLINK_ON 170
4.26.2.7 LCD_CNF_CURSOR_OFF 170
4.26.2.8 LCD_CNF_CURSOR_ON 170
4.26.2.9 LCD_CNF_DISPLAY_OFF 170
4.26.2.10 LCD_CNF_DISPLAY_ON 170
4.26.2.11 LCD_CNF_ENTRY_MODE_DIRECTION_LEFT 171
4.26.2.12 LCD_CNF_ENTRY_MODE_DIRECTION_RIGHT 171
4.26.2.13 LCD_CNF_ENTRY_MODE_DISPLAY_SHIFT_OFF 171
4.26.2.14 LCD_CNF_ENTRY_MODE_DISPLAY_SHIFT_ON 171
4.26.2.15 LCD_CNF_FONT_5_11 171
4.26.2.16 LCD_CNF_FONT_5_8 172
4.26.2.17 LCD_CNF_ONE_LINE 172

4.26.2.18 LCD_CNF_SHIFT_ID 172
4.26.2.19 LCD_CNF_SHIFT_SH 172
4.26.2.20 LCD_CNF_TWO_LINE 172
4.26.2.21 LCD_DISPLAY_CTRL_FIELD_B 173
4.26.2.22 LCD_DISPLAY_CTRL_FIELD_C 173
4.26.2.23 LCD_DISPLAY_CTRL_FIELD_D 173
4.26.2.24 LCD_FCT_SET_FIELD_DL 173
4.26.2.25 LCD_FCT_SET_FIELD_F 173
4.26.2.26 LCD_FCT_SET_FIELD_N 174
4.26.2.27 LCD_INST_CLR_DISPLAY_BIT 174
4.26.2.28 LCD_INST_DISPLAY_CTRL 174
4.26.2.29 LCD_INST_ENTRY_MODE_SET 174
4.26.2.30 LCD_INST_FUNCTION_SET 174
4.26.2.31 LCD_INST_SET_DDRAM_ADDR 175
4.26.2.32 LCD_RAM_1_LINE_MAX 175
4.26.2.33 LCD_RAM_1_LINE_MIN 175
4.26.2.34 LCD_RAM_2_LINES_MAX_1 175
4.26.2.35 LCD_RAM_2_LINES_MAX_2 175
4.26.2.36 LCD_RAM_2_LINES_MIN_1 176
4.26.2.37 LCD_RAM_2_LINES_MIN_2 176
4.26.2.38 LCD_SIZE_NB_CHAR_PER_LINE 176
4.26.2.39 LCD_SIZE_NB_LINES 176
4.26.2.40 LCD_WAIT_CLR_RETURN 176
4.26.2.41 LCD_WAIT_OTHER_MODES 177
4.26.2.42 RS_PIN 177
4.26.2.43 RW_PIN 177
4.26.3 Enumeration Type Documentation 177
4.26.3.1 T_LCD_command 177
4.26.3.2 T_LCD_config_mode 178
4.26.3.3 T_LCD_ram_area 178

4.27 LinkedList.cpp File Reference 178
4.28 LinkedList.h File Reference 179
4.28.1 Detailed Description 179
4.28.2 Typedef Documentation 180
4.28.2.1 CompareFctPtr_t 180
4.29 main.cpp File Reference 180
4.29.1 Detailed Description 181
4.29.2 Function Documentation 181
4.29.2.1 ISR() [1/2] 181
4.29.2.2 ISR() [2/2] 182
4.29.2.3 main() 182
4.30 main.h File Reference 183
4.30.1 Detailed Description 183
4.30.2 Macro Definition Documentation 184
4.30.2.1 DEBUG_MODE 184
4.31 operators.cpp File Reference 184
4.31.1 Detailed Description 184
4.31.2 Function Documentation 185
4.31.2.1 operator delete() 185
4.31.2.2 operator new() 185
4.32 operators.h File Reference 185
4.32.1 Detailed Description 186
4.32.2 Function Documentation 186
4.32.2.1 operator delete() 186
4.32.2.2 operator new() 187
4.33 scheduler.cpp File Reference 187
4.33.1 Detailed Description 188
4.33.2 Variable Documentation 188
4.33.2.1 p_scheduler 188
4.34 scheduler.h File Reference 188

4.34.1 Detailed Description 189
4.34.2 Macro Definition Documentation 189
4.34.2.1 PRESCALER_PERIODIC_TIMER 189
4.34.2.2 SW_PERIOD_MS 190
4.34.2.3 TIMER_CTC_VALUE 190
4.34.3 Typedef Documentation 190
4.34.3.1 TaskPtr_t 190
4.34.4 Variable Documentation 190
4.34.4.1 p_scheduler 190
4.35 String.cpp File Reference 191
4.35.1 Detailed Description 191
4.36 String.h File Reference 191
4.36.1 Detailed Description 192
4.37 TempSensor.cpp File Reference 192
4.37.1 Detailed Description 192
4.37.2 Macro Definition Documentation 193
4.37.2.1 PIT_BEFORE_INVALID 193
4.38 TempSensor.h File Reference 193
4.38.1 Detailed Description 193
4.38.2 Macro Definition Documentation 193
4.38.2.1 PERIOD_MS_TASK_TEMP_SENSOR 194
4.39 timer.cpp File Reference 194
4.39.1 Detailed Description 194
4.40 timer.h File Reference 194
4.40.1 Detailed Description 195
4.41 usart.cpp File Reference 195
4.41.1 Detailed Description 195
4.42 usart.h File Reference 196
4.42.1 Detailed Description 196

Index **197**

Chapter 1

Class Index

1.1 Class List

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

CpuLoad
Class defining CPU load libraries 5
DebugInterface
Class used for debugging on usart link 10
DebugManagement
Debug management class 17
dht22
DHT 22 driver class 22
dio
DIO class 25
DisplayInterface
Display interface services class 34
DisplayManagement
Display management class 46
I2C
Two-wire serial interface (I2C) class definition 51
keepAliveLed
Class for keep-alive LED blinking 55
LCD
Class for LCD S2004A display driver 57
LinkedList
Linked list class 74
scheduler
Scheduler class 81
String
String management class 90
T_ASW_cnf_struct
ASW configuration structure 99
T_BSW_cnf_struct
BSW configuration structure 101
T_display_data
Structure containing display data 103
T_Display_shift_data
Structure containing shift data 105
T_LCD_conf_struct
Structure defining LCD configuration 106

LinkedList::T_LL_element
Type defining a linked list element 109
scheduler::Task_t
Type defining a task structure 110
TempSensor
Class for temperature sensor 111
timer
Class defining a timer 122
usart
USART serial bus class 126

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

asw.cpp
ASW main file . . . 133
asw.h
ASW main header file . . . 135
bsw.cpp
BSW main file . . . 137
bsw.h
BSW main header file . . . 138
CpuLoad.cpp
Defines functions of class CpuLoad . . . 140
CpuLoad.h
CpuLoad class header file . . . 141
DebugInterface.cpp . . . 142
DebugInterface.h . . . 142
DebugManagement.cpp
Debug management class source file . . . 143
DebugManagement.h
Debug management class header file . . . 144
dht22.cpp
This file defines classes for DHT22 driver . . . 146
dht22.h
DHT22 driver header file . . . 147
dio.cpp
DIO library . . . 148
dio.h
DIO library header file . . . 148
dio_port_cnf.h
Digital ports configuration file . . . 151
dio_reg_atm2560.h . . . 152
DisplayInterface.cpp
Source code file for display services . . . 155
DisplayInterface.h
DisplayInterface class header file . . . 156
DisplayManagement.cpp
Display management source file . . . 158

DisplayManagement.h
Display management class header file . . . 159
I2C.cpp
Two-wire interface (I2C) source file . . . 162
I2C.h
I2C class header file . . . 163
keepAliveLed.cpp
Definition of function for class keepAliveLed . . . 164
keepAliveLed.h
Class keepAliveLed header file . . . 165
LCD.cpp
LCD class source file . . . 166
LCD.h
LCD class header file . . . 167
LinkedList.cpp . . . 178
LinkedList.h
Linked List library header file . . . 179
main.cpp
Background task file . . . 180
main.h
Background task header file . . . 183
operators.cpp
C++ operators definitions . . . 184
operators.h
C++ operators definitions header file . . . 185
scheduler.cpp
Defines scheduler class . . . 187
scheduler.h
Scheduler class header file . . . 188
String.cpp
String class source file . . . 191
String.h
String class header file . . . 191
TempSensor.cpp
Defines function of class TempSensor . . . 192
TempSensor.h
Class TempSensor header file . . . 193
timer.cpp
Defines function for class timer . . . 194
timer.h
Timer class header file . . . 194
usart.cpp
BSW library for USART . . . 195
usart.h
Header file for USART library . . . 196

Chapter 3

Class Documentation

3.1 CpuLoad Class Reference

Class defining CPU load libraries.

`#include <CpuLoad.h>`

Public Member Functions

- CpuLoad ()

  CpuLoad class constructor.
- void ComputeCPULoad ()

  Computes current CPU load.
- uint8_t getCurrrentCPULoad ()

  Get current CPU load value.
- uint8_t getAverageCPULoad ()

  Get average CPU load value.
- uint8_t getMaxCPULoad ()

  Get maximum CPU load value.

Private Attributes

- uint8_t current_load
- uint8_t avg_load
- uint8_t max_load
- uint8_t sample_cnt
- uint8_t sample_mem [NB_OF_SAMPLES]
- uint8_t sample_idx
- uint16_t last_sum_value

3.1.1 Detailed Description

Class defining CPU load libraries.

This class defines tools to compute and monitor CPU load.

Definition at line 19 of file CpuLoad.h.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 CpuLoad()

`CpuLoad::CpuLoad ( )`

CpuLoad class constructor.

This function initializes class CpuLoad

Returns

Nothing

Definition at line 13 of file CpuLoad.cpp.

3.1.3 Member Function Documentation

3.1.3.1 ComputeCPULoad()

`void CpuLoad::ComputeCPULoad ( )`

Computes current CPU load.

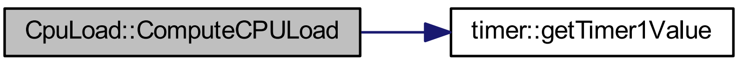
This function computes the current CPU load using value of the timer used by the scheduler at the end of the periodic cycle. This value is divided by the PIT period to obtain CPU load;

Returns

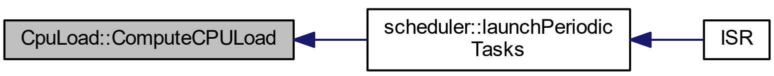
Nothing

Definition at line 27 of file CpuLoad.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.1.3.2 getAverageCPULoad()

```
uint8_t CpuLoad::getAverageCPULoad ( ) [inline]
```

Get average CPU load value.

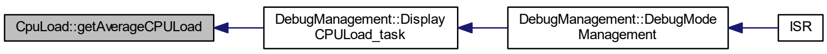
This function returns the average CPU load value

Returns

Average CPU load value

Definition at line 56 of file CpuLoad.h.

Here is the caller graph for this function:



3.1.3.3 getCurrrentCPULoad()

```
uint8_t CpuLoad::getCurrrentCPULoad ( ) [inline]
```

Get current CPU load value.

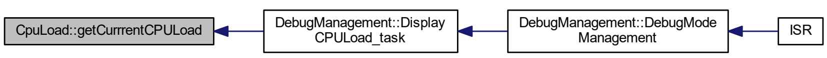
This function returns the current CPU load value

Returns

Current CPU load value

Definition at line 45 of file CpuLoad.h.

Here is the caller graph for this function:



3.1.3.4 getMaxCPULoad()

```
uint8_t CpuLoad::getMaxCPULoad ( ) [inline]
```

Get maximum CPU load value.

This function returns the maximum CPU load value

Returns

Maximum CPU load value

Definition at line 67 of file CpuLoad.h.

Here is the caller graph for this function:

3.1.4 Member Data Documentation

3.1.4.1 avg_load

```
uint8_t CpuLoad::avg_load [private]
```

Average CPU load based on the last 50 cycles

Definition at line 74 of file CpuLoad.h.

3.1.4.2 current_load

```
uint8_t CpuLoad::current_load [private]
```

Current CPU load (load of last cycle)

Definition at line 73 of file CpuLoad.h.

3.1.4.3 last_sum_value

```
uint16_t CpuLoad::last_sum_value [private]
```

Value of the last computed sum (it will reduce the number of samples to sum and speed up execution time)

Definition at line 79 of file CpuLoad.h.

3.1.4.4 max_load

```
uint8_t CpuLoad::max_load [private]
```

Maximum CPU load since power on

Definition at line 75 of file CpuLoad.h.

3.1.4.5 sample_cnt

```
uint8_t CpuLoad::sample_cnt [private]
```

Number of samples used to compute average load

Definition at line 76 of file CpuLoad.h.

3.1.4.6 sample_idx

```
uint8_t CpuLoad::sample_idx [private]
```

Current measurement index (used to memorize the current measure at the correct location in table)

Definition at line 78 of file CpuLoad.h.

3.1.4.7 sample_mem

```
uint8_t CpuLoad::sample_mem[NB_OF_SAMPLES] [private]
```

Memorization of the last NB_OF_SAMPLES measures

Definition at line 77 of file CpuLoad.h.

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

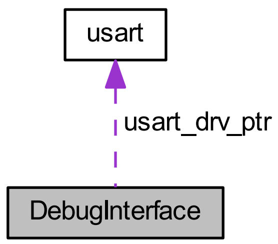
- CpuLoad.h
- CpuLoad.cpp

3.2 DebugInterface Class Reference

Class used for debugging on usart link.

`#include <DebugInterface.h>`

Collaboration diagram for DebugInterface:



Public Member Functions

- DebugInterface ()
  Class DebugInterface constructor.
- void sendInteger (uint16_t data, uint8_t base)
  Send a integer data on USART link.
- void sendBool (bool data, bool isText)
  Send a boolean data on USART link.
- void sendString (String *str)
  Send a string on USART link.
- void sendString (uint8_t *str)
  Send a chain of characters on USART link.
- void sendChar (uint8_t chr)
  Send a single character on USART link.
- uint8_t read ()
  USART read function.
- void nextLine ()
  Go to next line function.

Private Attributes

- usart * usart_drv_ptr

3.2.1 Detailed Description

Class used for debugging on usart link.

This class defines functions used for sending debug data on USART link.

Definition at line 21 of file DebugInterface.h.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 DebugInterface()

`DebugInterface::DebugInterface ( )`

Class DebugInterface constructor.

Initializes the class DebugInterface. It creates a new instance of USART driver of needed.

Returns

Nothing

Definition at line 43 of file DebugInterface.cpp.

3.2.3 Member Function Documentation

3.2.3.1 nextLine()

`void DebugInterface::nextLine ( )`

Go to next line function.

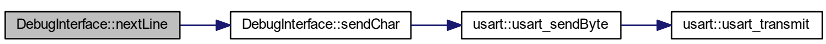
This function goes to the next line on the console display. It sends the two characters
and on the USART line.

Returns

Nothing

Definition at line 92 of file DebugInterface.cpp.

Here is the call graph for this function:



3.2.3.2 read()

```
uint8_t DebugInterface::read ( ) [inline]
```

USART read function.

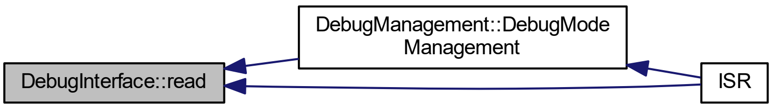
This function will read the last received byte on USART link

Returns

Received byte

Definition at line 82 of file DebugInterface.h.

Here is the caller graph for this function:



3.2.3.3 sendBool()

```
void DebugInterface::sendBool (
            bool data,
            bool isText )
```

Send a boolean data on USART link.

This function sends the requested boolean on USART link by calling driver's transmission function. The boolean data is first converted into a string and then sent. The parameter isText defines if the data is converted into a string (true/false) or an integer (1/0).

Parameters

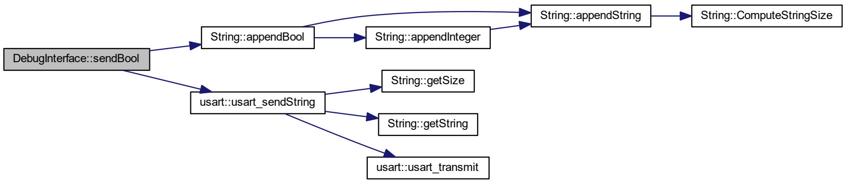
in	*data*	boolean data to be sent
in	*isText*	String conversion configuration

Returns

Nothing

Definition at line 83 of file DebugInterface.cpp.

Here is the call graph for this function:



3.2.3.4 sendChar()

```
void DebugInterface::sendChar (
            uint8_t chr )
```

Send a single character on USART link.

This function sends the requested character on USART link by calling driver's transmission function.

Parameters

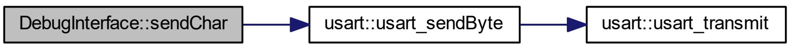
in	*chr*	Character to send.

Returns

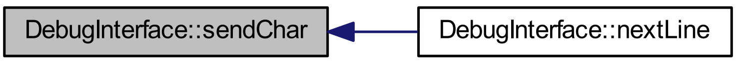
Nothing

Definition at line 65 of file DebugInterface.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.2.3.5 sendInteger()

```
void DebugInterface::sendInteger (
            uint16_t data,
            uint8_t base )
```

Send a integer data on USART link.

This function sends the requested integer on USART link by calling driver's transmission function. The integer is first converted into a string and then sent

Parameters

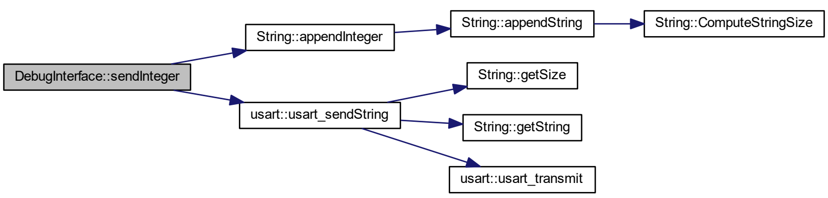
in	*data*	integer data to be sent
in	*base*	numerical base used to convert integer into string (between 2 and 36)

Returns

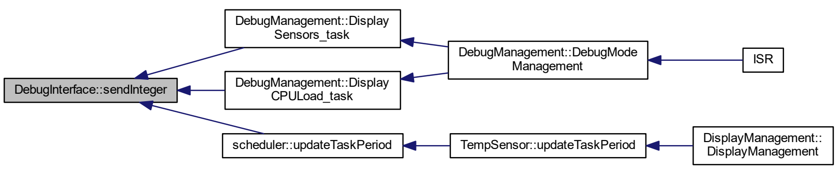
Nothing

Definition at line 70 of file DebugInterface.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.2.3.6 sendString() [1/2]

```
void DebugInterface::sendString (
            String * str )
```

Send a string on USART link.

This function sends the requested string on USART link by calling driver's transmission function

Parameters

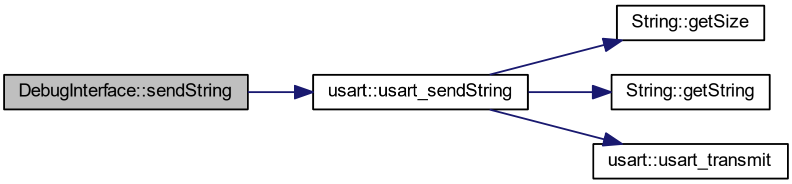
`in`	*str*	Pointer to the string being sent

Returns

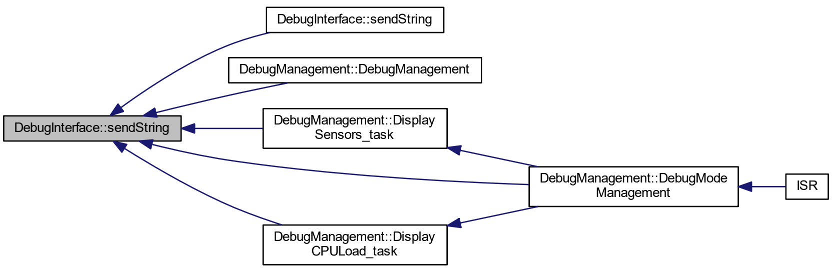
Nothing

Definition at line 52 of file DebugInterface.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.2.3.7 sendString() [2/2]

```
void DebugInterface::sendString (
            uint8_t * str )
```

Send a chain of characters on USART link.

This function sends the requested chain of characters on USART link by calling driver's transmission function. The chain is first converted into a string object.

Parameters

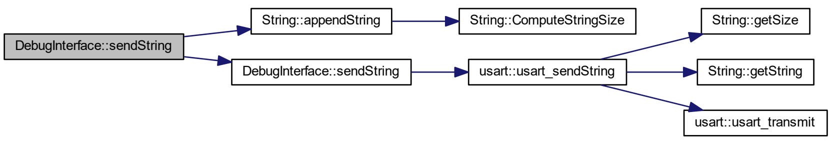
in	*str*	Pointer to the chain to send.

Returns

Nothing

Definition at line 58 of file DebugInterface.cpp.

Here is the call graph for this function:



3.2.4 Member Data Documentation

3.2.4.1 usart_drv_ptr

```
usart* DebugInterface::usart_drv_ptr  [private]
```

Pointer to USART driver object

Definition at line 99 of file DebugInterface.h.

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

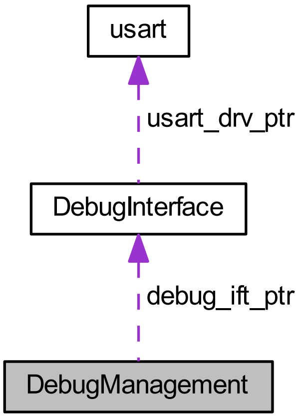
- DebugInterface.h
- DebugInterface.cpp

3.3 DebugManagement Class Reference

Debug management class.

```
#include <DebugManagement.h>
```

Collaboration diagram for DebugManagement:



Public Member Functions

- DebugManagement ()
  Class constructor.
- bool DebugModeManagement ()
  Management of debug mode.
- DebugInterface ∗ getIftPtr ()
  Interface pointer get function.

Static Public Member Functions

- static void DisplaySensors_task ()
  Displays sensors data on usart link.
- static void DisplayCPULoad_task ()
  Displays CPU load data on usart link.

Private Attributes

- DebugInterface ∗ debug_ift_ptr
- debug_state_t debug_state

3.3.1 Detailed Description

Debug management class.

This class manages the debug menu available on USART interface. It allows to display SW informations like sensors data, CPU load...

Definition at line 31 of file DebugManagement.h.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 DebugManagement()

`DebugManagement::DebugManagement ( )`

Class constructor.

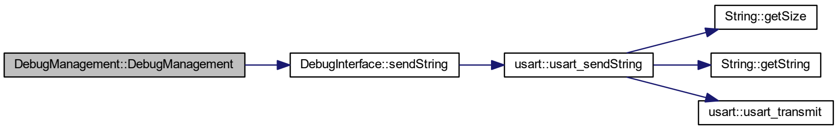
This function initializes the class. If needed, it creates a new instance of debug interface object.

Returns

Nothing

Definition at line 50 of file DebugManagement.cpp.

Here is the call graph for this function:



3.3.3 Member Function Documentation

3.3.3.1 DebugModeManagement()

`bool DebugManagement::DebugModeManagement ( )`

Management of debug mode.

This function manages the debug mode according to the following state machine :

- INIT state : handles user choice in main menu and selects next state
- DISPLAY_DATA state : display sensor data periodically
- DISPLAY CPU LOAD : display CPU load periodically

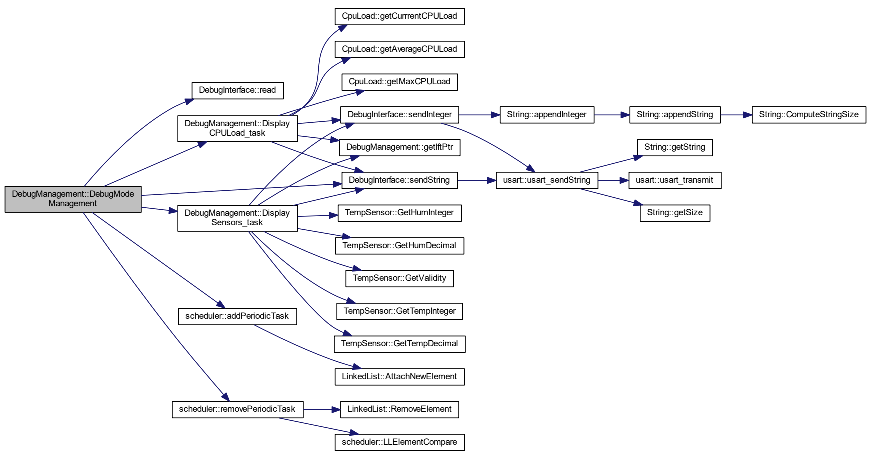
  It is called each time a data is received on USART and debug mode is active.

Returns

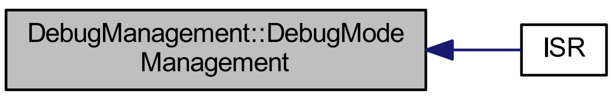
True if the debug mode shall be closed, false otherwise

Definition at line 112 of file DebugManagement.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.3.3.2 DisplayCPULoad_task()

```
void DebugManagement::DisplayCPULoad_task ( ) [static]
```

Displays CPU load data on usart link.

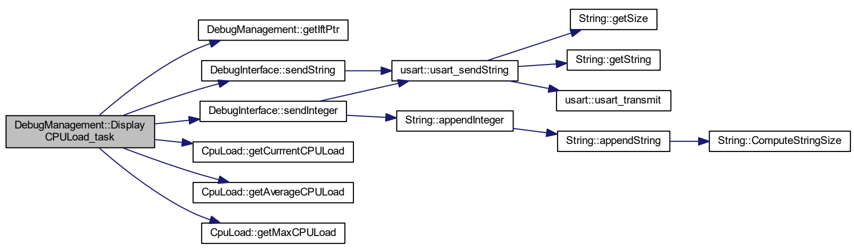
This task sends CPU load data (current and average load) on usart link every 5 seconds

Returns

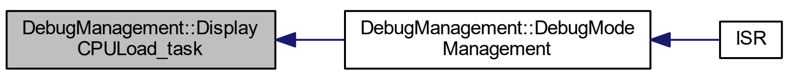
Nothing

Definition at line 98 of file DebugManagement.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.3.3.3 DisplaySensors_task()

```
void DebugManagement::DisplaySensors_task ( ) [static]
```

Displays sensors data on usart link.

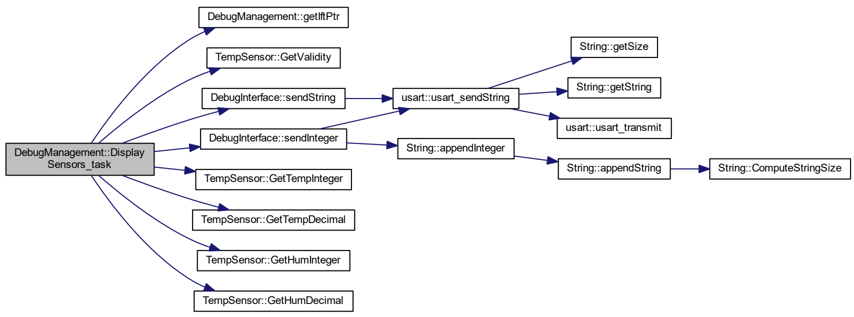
This task sends sensors data (temperature and humidity) on usart link every 5 seconds

Returns

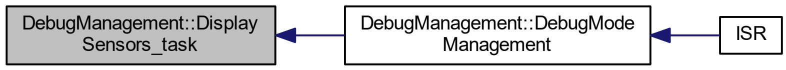
Nothing

Definition at line 65 of file DebugManagement.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.3.3.4 getIftPtr()

```
DebugInterface* DebugManagement::getIftPtr ( ) [inline]
```

Interface pointer get function.

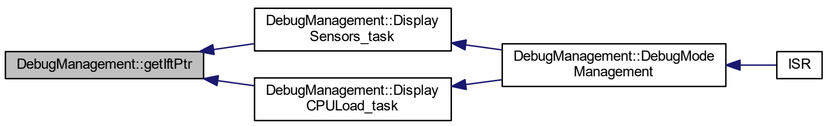
This function returns the pointer to the debug interface object

Returns

Pointer to debug interface

Definition at line 76 of file DebugManagement.h.

Here is the caller graph for this function:



3.3.4 Member Data Documentation

3.3.4.1 debug_ift_ptr

`DebugInterface* DebugManagement::debug_ift_ptr [private]`

Pointer to the debug interface object, which is used to send data on usart link

Definition at line 83 of file DebugManagement.h.

3.3.4.2 debug_state

`debug_state_t DebugManagement::debug_state [private]`

Current debug state

Definition at line 85 of file DebugManagement.h.

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

- DebugManagement.h
- DebugManagement.cpp

3.4 dht22 Class Reference

DHT 22 driver class.

`#include <dht22.h>`

Public Member Functions

- dht22 ()
  dht22 class constructor
- bool read (uint16_t *raw_humidity, uint16_t *raw_temperature)
  Reads the data from DHT22.

Private Member Functions

- void initializeCommunication ()
  Initializes the communication.

3.4.1 Detailed Description

DHT 22 driver class.

This class defines all useful functions for DHT22 temperature and humidity sensor

Definition at line 22 of file dht22.h.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 dht22()

`dht22::dht22 ( )`

dht22 class constructor

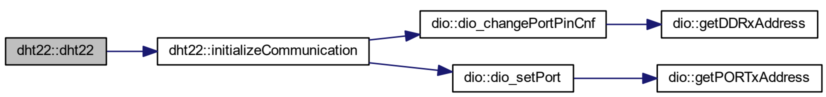
Initializes the class dht22

Returns

Nothing

Definition at line 22 of file dht22.cpp.

Here is the call graph for this function:



3.4.3 Member Function Documentation

3.4.3.1 initializeCommunication()

```
void dht22::initializeCommunication ( ) [private]
```

Initializes the communication.

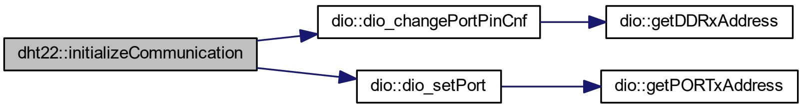
This function initializes the communication with DHT22 using 1-wire protocol

Returns

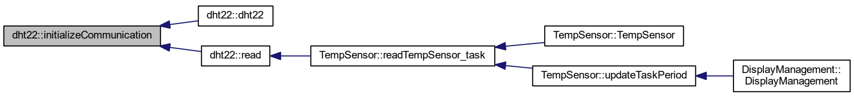
Nothing

Definition at line 198 of file dht22.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.4.3.2 read()

```
bool dht22::read (
            uint16_t * raw_humidity,
            uint16_t * raw_temperature )
```

Reads the data from DHT22.

This function communicates with DHT22 using 1-wire protocol to read raw values of temperature and humidity. A checksum check is done when communication is finished to validate the received data

Parameters

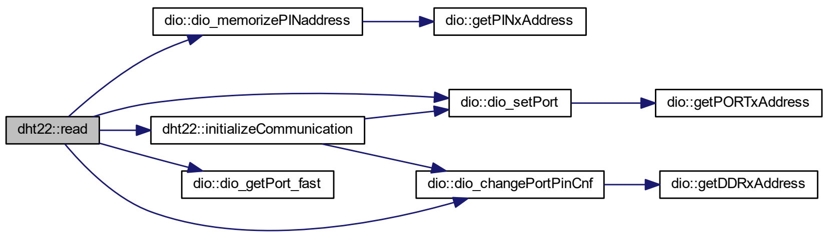
out	*raw_humidity*	Raw humidity value received from sensor
out	*raw_temperature*	Raw temperature value received from sensor

Returns

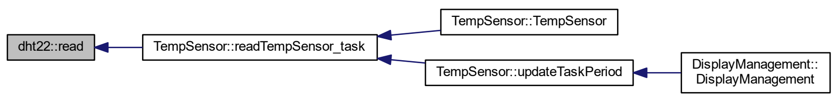
Validity of the read value

Definition at line 27 of file dht22.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

- dht22.h
- dht22.cpp

3.5 dio Class Reference

DIO class.

`#include <dio.h>`

Public Member Functions

- dio ()

  dio class constructor
- void dio_setPort (uint8_t portcode, bool state)

  Port setting function.
- void dio_invertPort (uint8_t portcode)

  Inverts the state of output port.
- bool dio_getPort (uint8_t portcode)

  Gets the logical state of selected pin.
- bool dio_getPort_fast (void)

  Gets the logical state of the memorized pin.
- void dio_changePortPinCnf (uint8_t portcode, uint8_t cnf)

  Changes the IO configuration of the selected pin.
- void dio_memorizePINaddress (uint8_t portcode)

  Memorizes PINx register address and pin index.

Private Member Functions

- void ports_init ()

    Digital ports hardware initialization function.
- uint8_t ∗ getPORTxAddress (uint8_t portcode)

    Gets the physical address of the requested register PORTx.
- uint8_t ∗ getPINxAddress (uint8_t portcode)

    Gets the physical address of the requested register PINx.
- uint8_t ∗ getDDRxAddress (uint8_t portcode)

    Gets the physical address of the requested register DDRx.

Private Attributes

- uint8_t ∗ PINx_addr_mem
- uint8_t PINx_idx_mem

3.5.1 Detailed Description

DIO class.

This class defines all useful functions for digital input/output ports

Definition at line 31 of file dio.h.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 dio()

```
dio::dio ( )
```

dio class constructor

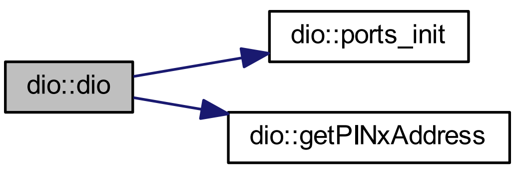
Initializes class dio and calls DIO hardware initialization function

Returns

Nothing

Definition at line 112 of file dio.cpp.

Here is the call graph for this function:



3.5.3 Member Function Documentation

3.5.3.1 dio_changePortPinCnf()

```
void dio::dio_changePortPinCnf (
            uint8_t portcode,
            uint8_t cnf )
```

Changes the IO configuration of the selected pin.

This function configures the selected pin as input or output according to parameter cnf. The corresponding port and pin index is extracted from parameter portcode.

Parameters

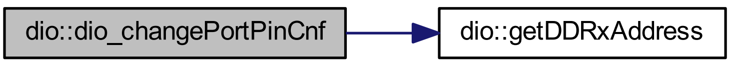
in	*portcode*	Encoded pin and register index
in	*cnf*	Requested configuration for the selected pin PORT_CNF_OUT (1) : pin configured as output PORT_CNF_IN (0) : pin configured as input

Returns

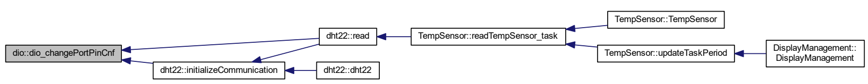
Nothing

Definition at line 149 of file dio.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.5.3.2 dio_getPort()

```
bool dio::dio_getPort (
            uint8_t portcode )
```

Gets the logical state of selected pin.

This function gets the logical value of the selected pin. The corresponding port and pin index is extracted from parameter portcode.

Parameters

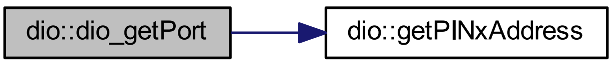
in	*portcode*	Encoded pin and register index

Returns

Logical state of selected pin

Definition at line 139 of file dio.cpp.

Here is the call graph for this function:



3.5.3.3 dio_getPort_fast()

```
bool dio::dio_getPort_fast (
            void )
```

Gets the logical state of the memorized pin.

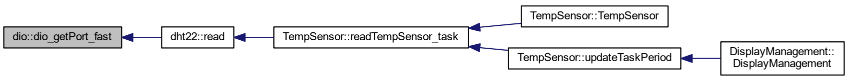
This function gets the logical value of the memorized pin. The corresponding port and pin index are stored in class members PINx_addr_mem and PINx_idx_mem. This mechanism is used to speed up reading time as this function no longer needs to extract register address and pin index from portcode.

Returns

Logical state of selected pin

Definition at line 171 of file dio.cpp.

Here is the caller graph for this function:



3.5.3.4 dio_invertPort()

```
void dio::dio_invertPort (
             uint8_t portcode )
```

Inverts the state of output port.

This function inverts the state of the chosen pin. The corresponding port and pin index is extracted from parameter portcode.

Parameters

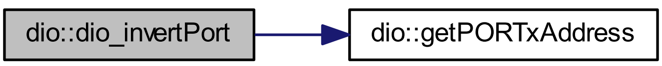
in	*portcode*	Encoded pin and register index

Returns

Nothing

Definition at line 131 of file dio.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

3.5.3.5 dio_memorizePINaddress()

```
void dio::dio_memorizePINaddress (
            uint8_t portcode )
```

Memorizes PINx register address and pin index.

This function is used to speed up reading of register PINx. Register address and pin index are decoded from portcode parameter and stored for later use by function dio_getPort_fast.

Parameters

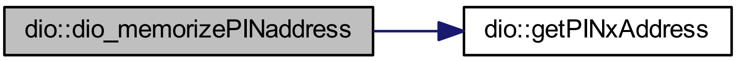
in	*portcode*	Encoded pin and register index

Returns

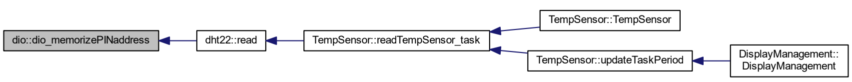
Nothing

Definition at line 165 of file dio.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.5.3.6 dio_setPort()

```
void dio::dio_setPort (
            uint8_t portcode,
            bool state )
```

Port setting function.

This function sets the requested digital output to the requested state. The corresponding port and pin index is extracted from parameter portcode.

Parameters

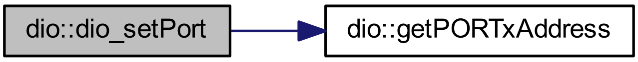
in	*portcode*	Encoded pin and register index
in	*state*	Requested state to set pin

Returns

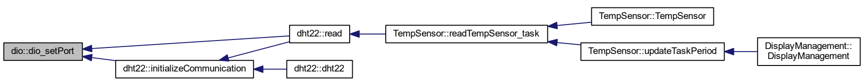
Nothing

Definition at line 121 of file dio.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.5.3.7 getDDRxAddress()

```
uint8_t * dio::getDDRxAddress (
            uint8_t portcode ) [private]
```

Gets the physical address of the requested register DDRx.

This function retrieves the address of the register DDRx where x is encoded into the parameter portcode.

Parameters

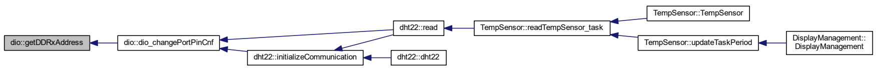
in	*portcode*	Encoded port code

Returns

Pointer to the DDRx register

Definition at line 83 of file dio.cpp.

Here is the caller graph for this function:



3.5.3.8 getPINxAddress()

```
uint8_t * dio::getPINxAddress (
            uint8_t portcode )  [private]
```

Gets the physical address of the requested register PINx.

This function retrieves the address of the register PINx where x is encoded into the parameter portcode.

Parameters

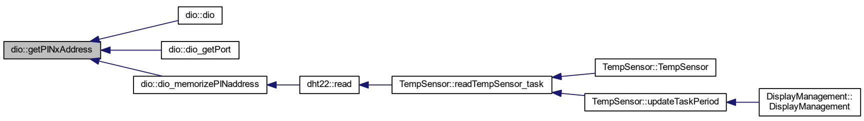
in	*portcode*	Encoded port code

Returns

Pointer to the PINx register

Definition at line 54 of file dio.cpp.

Here is the caller graph for this function:



3.5.3.9 getPORTxAddress()

```
uint8_t * dio::getPORTxAddress (
             uint8_t portcode )  [private]
```

Gets the physical address of the requested register PORTx.

This function retrieves the address of the register PORTx where x is encoded into the parameter portcode.

Parameters

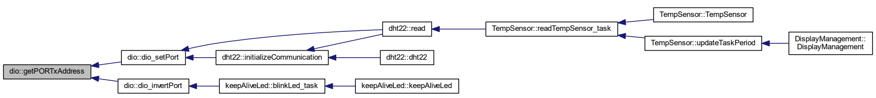
in	*portcode*	Encoded port code

Returns

Pointer to the PORTx register

Definition at line 25 of file dio.cpp.

Here is the caller graph for this function:



3.5.3.10 ports_init()

```
void dio::ports_init ( )  [private]
```

Digital ports hardware initialization function.

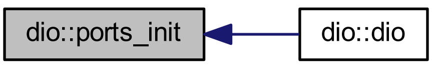
This function initializes digital ports as input or output and sets their initial values

Returns

Nothing

Definition at line 16 of file dio.cpp.

Here is the caller graph for this function:



3.5.4 Member Data Documentation

3.5.4.1 PINx_addr_mem

`uint8_t* dio::PINx_addr_mem [private]`

Memorizes physical address of register PINx in order to speed up register reading time in function dio_getPort_fast

Definition at line 146 of file dio.h.

3.5.4.2 PINx_idx_mem

`uint8_t dio::PINx_idx_mem [private]`

Memorizes pin index of register PINx in order to speed up register reading time in function dio_getPort_fast

Definition at line 147 of file dio.h.

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

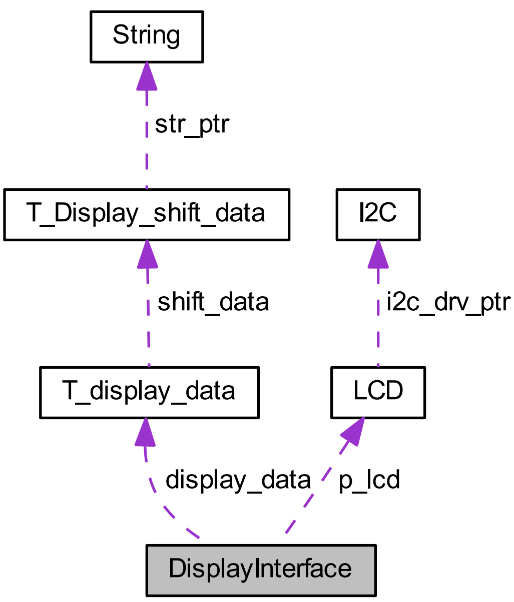
- dio.h
- dio.cpp

3.6 DisplayInterface Class Reference

Display interface services class.

`#include <DisplayInterface.h>`

Collaboration diagram for DisplayInterface:



Public Member Functions

- DisplayInterface (const T_LCD_conf_struct *LCD_init_cnf)

    Class constructor.
- bool DisplayFullLine (uint8_t *str, uint8_t size, uint8_t line, T_DisplayInterface_LineDisplayMode mode=NORMAL, T_DisplayInterface_LineAlignment alignment=LEFT)

    Line display function.
- bool ClearLine (uint8_t line)

    Line clearing function.
- void ClearFullScreen ()

    Screen cleaning function.
- bool IsLineEmpty (uint8_t line)

    Empty line get function.
- T_display_data * getDisplayDataPtr ()

    Display data get function.
- void setLineAlignmentAndRefresh (uint8_t line, T_DisplayInterface_LineAlignment alignment)

    Text alignment function.

Static Public Member Functions

- static void shiftLine_task ()

    Line shifting periodic task.

Private Member Functions

- uint8_t FindFirstCharAddr (uint8_t line)

    Finds start address of a line.
- void RefreshLine (uint8_t line)

    Line refresh function.
- void ClearStringInDataStruct (uint8_t line)

    String data clearing structure.
- void setLineAlignment (uint8_t line)

    Text alignment setting function.
- void updateLineAndRefresh (uint8_t *str, uint8_t size, uint8_t line)

    Line data string update function.

Private Attributes

- LCD * p_lcd
- uint32_t dummy
- T_display_data display_data [LCD_SIZE_NB_LINES]
- bool isShiftInProgress

3.6.1 Detailed Description

Display interface services class.

This class defines the services used for interfacing display management services and LCD screen driver

Definition at line 76 of file DisplayInterface.h.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 DisplayInterface()

```
DisplayInterface::DisplayInterface (
            const T_LCD_conf_struct * LCD_init_cnf )
```

Class constructor.

This function initializes all class variables and instantiates the LCD driver according to the given configuration.

Parameters

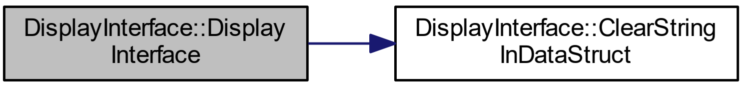
in	*LCD_init_cnf*	Initial configuration of the screen

Returns

Nothing

Definition at line 40 of file DisplayInterface.cpp.

Here is the call graph for this function:



3.6.3 Member Function Documentation

3.6.3.1 ClearFullScreen()

```
void DisplayInterface::ClearFullScreen ( )
```

Screen cleaning function.

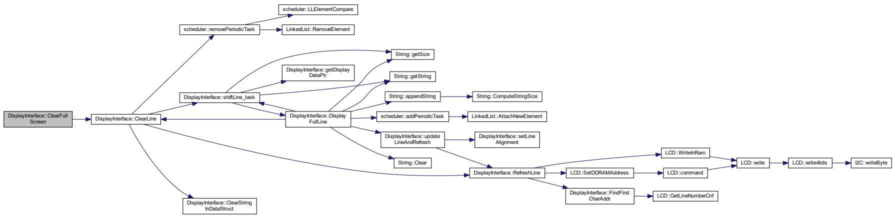
This functions clears the entire display. It uses the ClearLine function on every line of the screen.

Returns

Nothing

Definition at line 278 of file DisplayInterface.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

3.6.3.2 ClearLine()

```
bool DisplayInterface::ClearLine (
            uint8_t line )
```

Line clearing function.

This function clears the requested line. It sets the corresponding DDRAM addresses to the ASCII value of space character. If it was the last line with a display shift in progress, it removes the periodic task from the scheduler.

Parameters

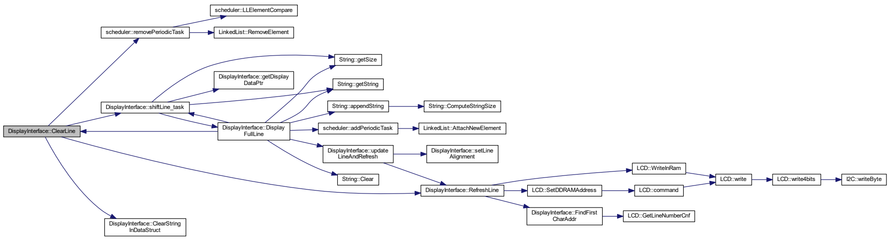
in	*line*	Line to clear

Returns

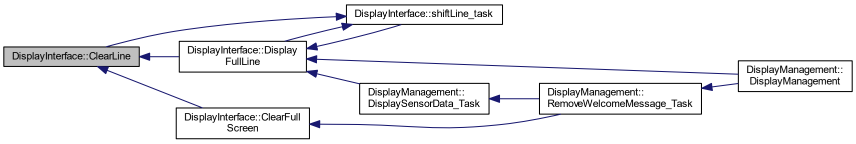
True if the line has been cleared, false otherwise

Definition at line 234 of file DisplayInterface.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.6.3.3 ClearStringInDataStruct()

```
void DisplayInterface::ClearStringInDataStruct (
            uint8_t line )  [private]
```

String data clearing structure.

This function clears the string contained in the display data structure. It sets all characters to space character.

Parameters

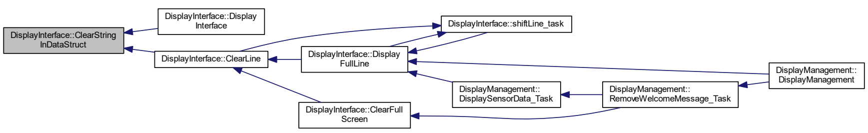
in	*line*	Line to clear

Returns

Nothing

Definition at line 187 of file DisplayInterface.cpp.

Here is the caller graph for this function:



3.6.3.4 DisplayFullLine()

```
bool DisplayInterface::DisplayFullLine (
            uint8_t * str,
            uint8_t size,
            uint8_t line,
            T_DisplayInterface_LineDisplayMode mode = NORMAL,
            T_DisplayInterface_LineAlignment alignment = LEFT )
```

Line display function.

This function displays the given string on the requested line. If the string is too long to be displayed entirely, the behavior is defined by the selected mode.

Parameters

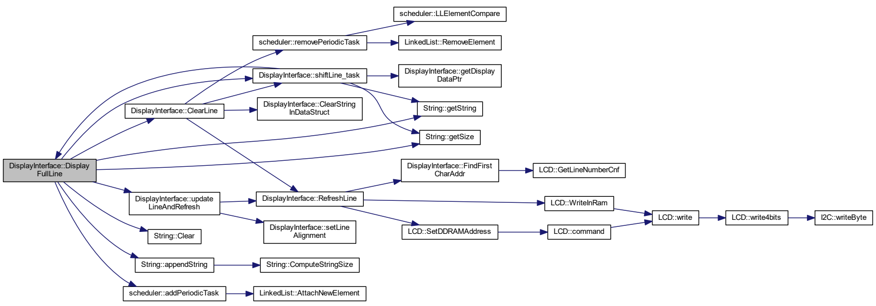
in	*str*	Pointer to the string to display
in	*size*	Size of the string to display
in	*line*	Index of the line where the string shall be displayed
in	*mode*	Display mode
in	*alignment*	Requested alignment for the line

Returns

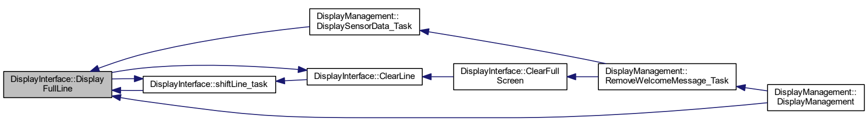
True if the line has been correctly displayed, false otherwise

Definition at line 72 of file DisplayInterface.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.6.3.5 FindFirstCharAddr()

```
uint8_t DisplayInterface::FindFirstCharAddr (
            uint8_t line ) [private]
```

Finds start address of a line.

This function finds the address in DDRAM of the first character of a line.

Parameters

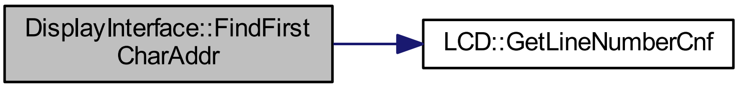
in	*line*	Line which address shall be found

Returns

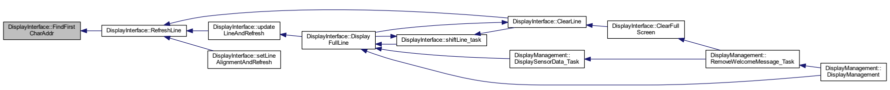
Address in DDRAM of the first character of the line

Definition at line 195 of file DisplayInterface.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.6.3.6 getDisplayDataPtr()

```
T_display_data* DisplayInterface::getDisplayDataPtr ( ) [inline]
```

Display data get function.

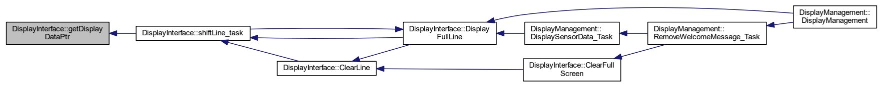
This function returns a pointer to the display data structure.

Returns

Pointer to display data structure.

Definition at line 142 of file DisplayInterface.h.

Here is the caller graph for this function:



3.6.3.7 IsLineEmpty()

```
bool DisplayInterface::IsLineEmpty (
            uint8_t line )
```

Empty line get function.

This function answers if the line given in parameter is empty or not, according to the table isLineEmpty[]

Parameters

in	*line*	Requested line

Returns

True if the line is empty, false otherwise

Definition at line 286 of file DisplayInterface.cpp.

3.6.3.8 RefreshLine()

```
void DisplayInterface::RefreshLine (
            uint8_t line ) [private]
```

Line refresh function.

This function refreshes the display on the requested line. It computes the screen RAM address and writes the string to display into the screen RAM. It shall be called everytime the string in display data structure is updated.

Parameters

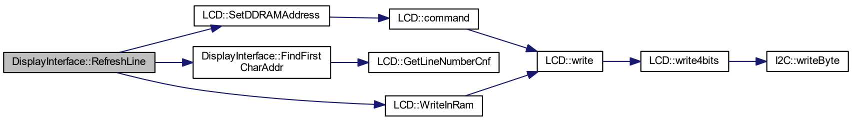
in	*line*	Line to refresh

Returns

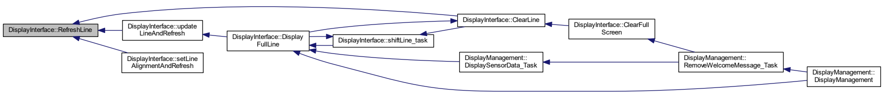
Nothing

Definition at line 174 of file DisplayInterface.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.6.3.9 setLineAlignment()

```
void DisplayInterface::setLineAlignment (
            uint8_t line )  [private]
```

Text alignment setting function.

This function updates the text alignment on the requested line. The string in the data structure is updated with the new alignment. The alignment parameter in the data structure shall be updated before calling this function.

Parameters

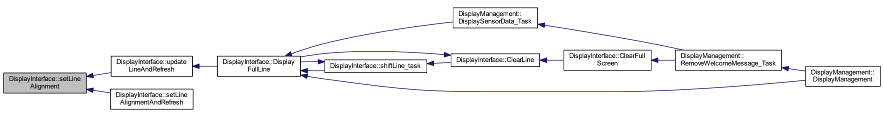
in	*line*	Line to update

Returns

Nothing

Definition at line 343 of file DisplayInterface.cpp.

Here is the caller graph for this function:



3.6.3.10 setLineAlignmentAndRefresh()

```
void DisplayInterface::setLineAlignmentAndRefresh (
            uint8_t line,
            T_DisplayInterface_LineAlignment alignment )
```

Text alignment function.

This function updates the text alignment on the requested line. It calls the private function setLineAlignment to update data structure and then refreshes the display. Nothing is done if the requested alignment is the same than the current one, if the line is empty or if the line is in line shift mode.

Parameters

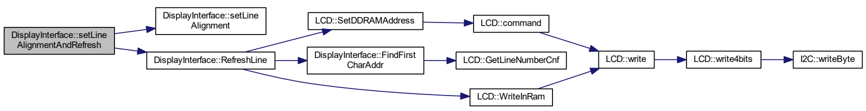
in	*line*	Requested line to update
in	*alignment*	Requested alignment for the text

Returns

Nothing

Definition at line 461 of file DisplayInterface.cpp.

Here is the call graph for this function:



3.6.3.11 shiftLine_task()

```
void DisplayInterface::shiftLine_task ( )  [static]
```

Line shifting periodic task.

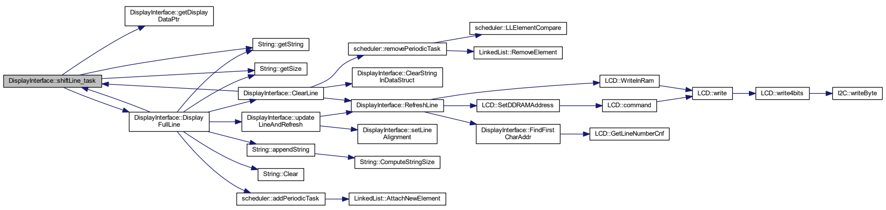
This function is called periodically by the scheduler. It shifts all the lines in line shifting mode and updates the data structures.

Returns

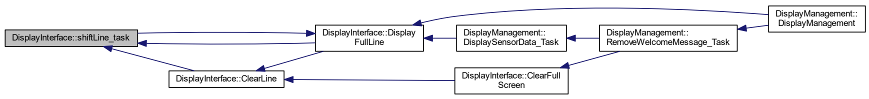
Nothing

Definition at line 295 of file DisplayInterface.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.6.3.12 updateLineAndRefresh()

```
void DisplayInterface::updateLineAndRefresh (
            uint8_t * str,
            uint8_t size,
            uint8_t line )  [private]
```

Line data string update function.

This function updates the data string and refreshes the display. The string is aligned according to the given parameter.

Parameters

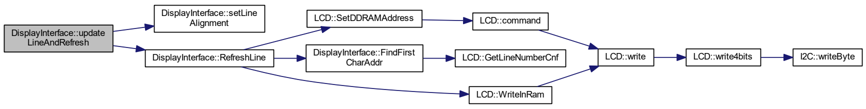
in	*str*	Pointer to the string to display
in	*size*	Size of the string
in	*line*	Line to update

Returns

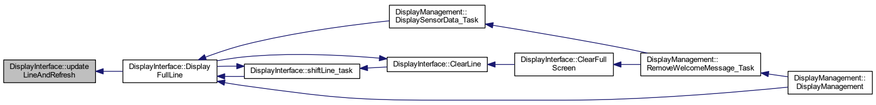
Nothing

Definition at line 158 of file DisplayInterface.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.6.4 Member Data Documentation

3.6.4.1 display_data

`T_display_data DisplayInterface::display_data[LCD_SIZE_NB_LINES] [private]`

Screen display data

Definition at line 165 of file DisplayInterface.h.

3.6.4.2 dummy

`uint32_t DisplayInterface::dummy [private]`

Needed for data alignment

Definition at line 164 of file DisplayInterface.h.

3.6.4.3 isShiftInProgress

`bool DisplayInterface::isShiftInProgress [private]`

Flag indicating if a shift is in progress on any line

Definition at line 166 of file DisplayInterface.h.

3.6.4.4 p_lcd

```
LCD* DisplayInterface::p_lcd [private]
```

Pointer to the attached LCD driver object

Definition at line 163 of file DisplayInterface.h.

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

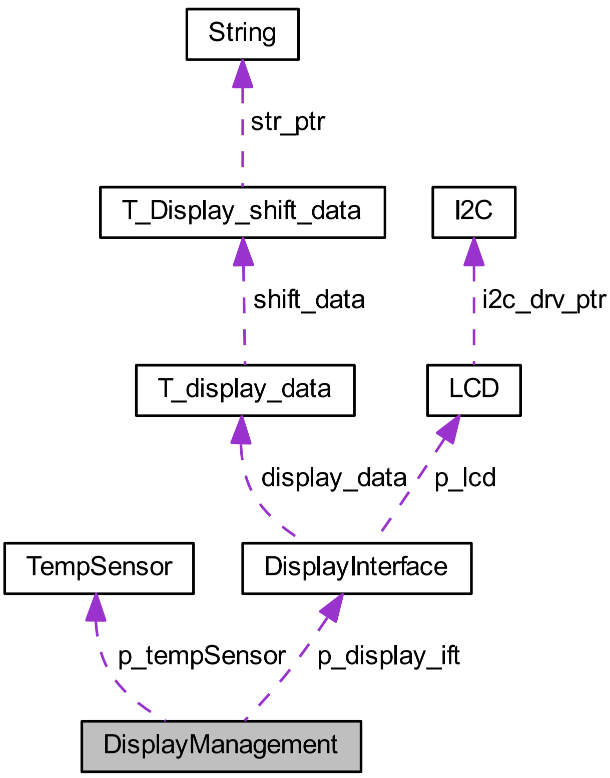
- DisplayInterface.h
- DisplayInterface.cpp

3.7 DisplayManagement Class Reference

Display management class.

`#include <DisplayManagement.h>`

Collaboration diagram for DisplayManagement:



Public Member Functions

- DisplayManagement ()
  Class constructor.
- DisplayInterface ∗ GetIftPointer ()
  Interface pointer get function.
- TempSensor ∗ GetTempSensorPtr ()
  Sensor pointer get function.

Static Public Member Functions

- static void DisplaySensorData_Task ()
  Periodic task for displaying sensor data.
- static void RemoveWelcomeMessage_Task ()
  End of welcome message task.

Private Attributes

- DisplayInterface ∗ p_display_ift
- TempSensor ∗ p_tempSensor

3.7.1 Detailed Description

Display management class.

This class manages all displays. It is a top-level class. It retrieves the data computed by other ASW classes and displays them. It is interfaced with DisplayInterface class to display data on screens. One interface class is used for each screen.

Definition at line 52 of file DisplayManagement.h.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 DisplayManagement()

```
DisplayManagement::DisplayManagement ( )
```

Class constructor.

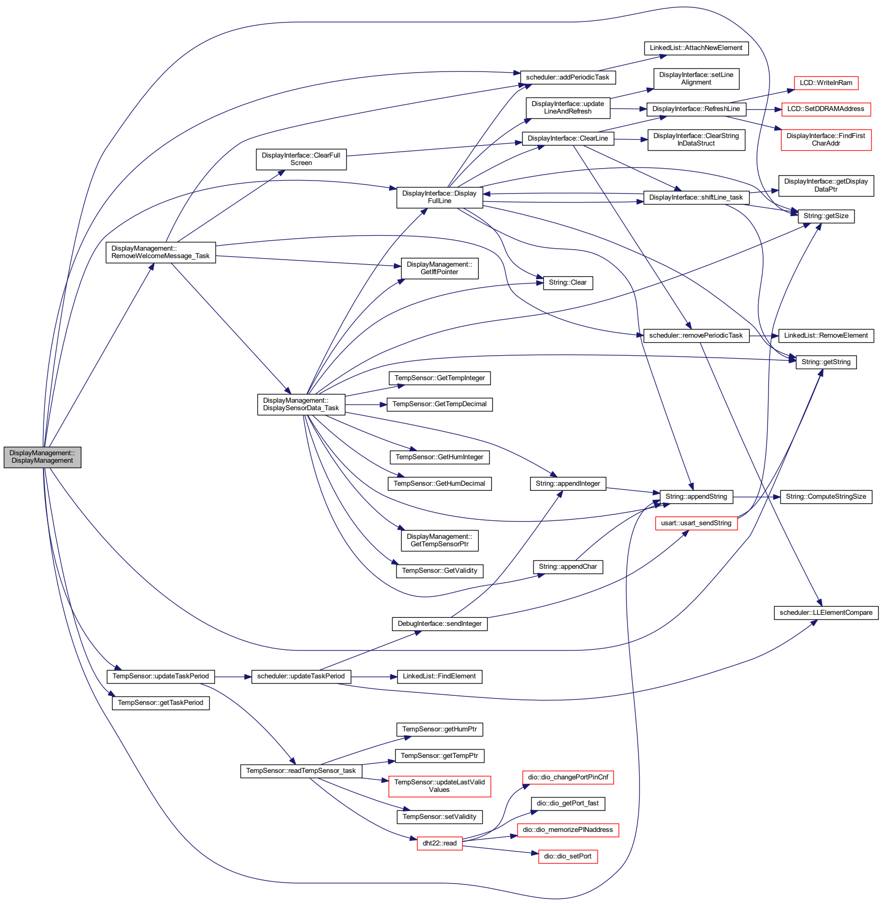
This class initializes display management.
It created a display interface object and initializes all class variables.

Returns

Nothing

Definition at line 38 of file DisplayManagement.cpp.

Here is the call graph for this function:



3.7.3 Member Function Documentation

3.7.3.1 DisplaySensorData_Task()

```
void DisplayManagement::DisplaySensorData_Task ( )  [static]
```

Periodic task for displaying sensor data.

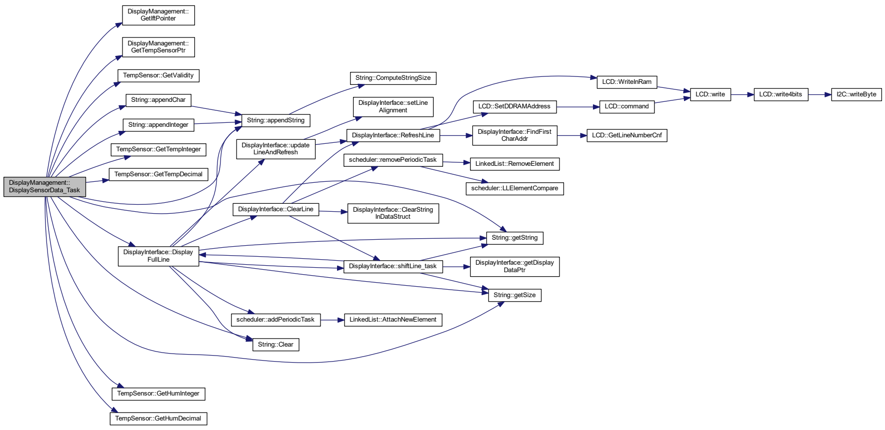
This function displays the sensors data on the screen. Currently temperature and humidity data coming from dht22 sensor are displayed.
It is called periodically by scheduler.

Returns

Nothing

Definition at line 79 of file DisplayManagement.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

3.7.3.2 GetIftPointer()

```
DisplayInterface* DisplayManagement::GetIftPointer ( ) [inline]
```

Interface pointer get function.

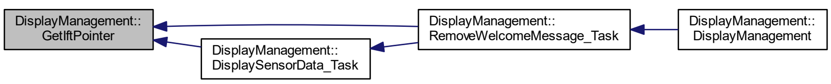
This function returns the pointer to the display interface object

Returns

Pointer to display interface object

Definition at line 81 of file DisplayManagement.h.

Here is the caller graph for this function:



3.7.3.3 GetTempSensorPtr()

`TempSensor* DisplayManagement::GetTempSensorPtr ( ) [inline]`

Sensor pointer get function.

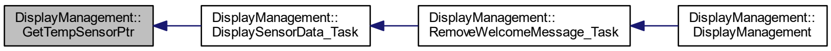
This function returns the pointer to the temperature sensor object

Returns

Pointer to sensor object

Definition at line 92 of file DisplayManagement.h.

Here is the caller graph for this function:



3.7.3.4 RemoveWelcomeMessage_Task()

`void DisplayManagement::RemoveWelcomeMessage_Task ( ) [static]`

End of welcome message task.

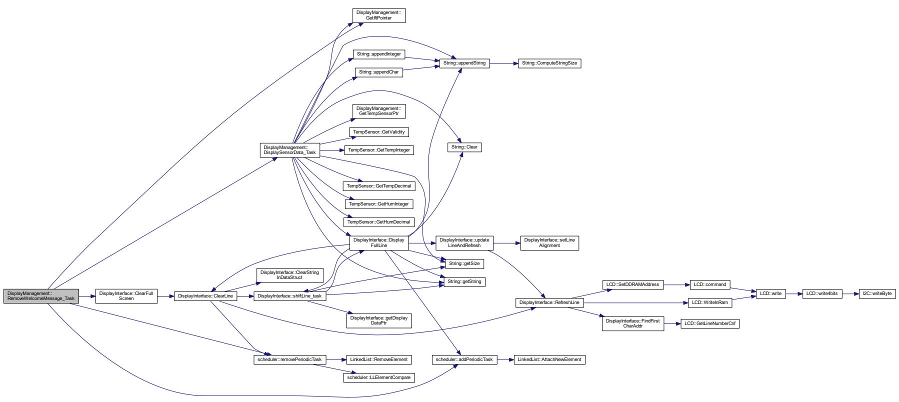
This task clears the welcome message from the screen and start periodic display of sensor data. This task shall be added in scheduler when the welcome message is displayed on screen. As it shall be called only once, the task removes itself from the scheduler after the first call.

Returns

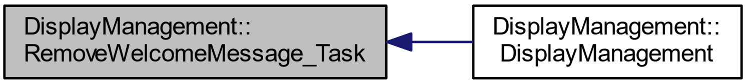
Nothing

Definition at line 65 of file DisplayManagement.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.7.4 Member Data Documentation

3.7.4.1 p_display_ift

```
DisplayInterface* DisplayManagement::p_display_ift  [private]
```

Pointer to the display interface object

Definition at line 109 of file DisplayManagement.h.

3.7.4.2 p_tempSensor

```
TempSensor* DisplayManagement::p_tempSensor  [private]
```

Pointer to the temperature sensor object

Definition at line 110 of file DisplayManagement.h.

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

- DisplayManagement.h
- DisplayManagement.cpp

3.8 I2C Class Reference

Two-wire serial interface (I2C) class definition.

```
#include <I2C.h>
```

Public Member Functions

- I2C (uint32_t l_bitrate)

    I2C class constructor.
- bool writeByte (uint8_t ∗data)

    Byte sending function.
- void setTxAddress (uint8_t address)

    Setting function for Tx I2C address.
- void setBitRate (uint32_t l_bitrate)

    Variable bitrate setting function.

Private Member Functions

- void initializeBus ()

    I2C bus initialization.

Private Attributes

- uint8_t tx_address
- uint32_t bitrate

3.8.1 Detailed Description

Two-wire serial interface (I2C) class definition.

This class manages I2C driver.

Definition at line 23 of file I2C.h.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 I2C()

```
I2C::I2C (
            uint32_t l_bitrate )
```

I2C class constructor.

This function initializes the I2C class and calls bus initialization function

Parameters

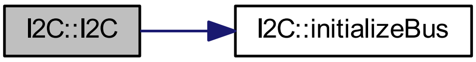
in	*l_bitrate*	Requested bitrate for I2C bus (in Hz)

Returns

Nothing

Definition at line 15 of file I2C.cpp.

Here is the call graph for this function:



3.8.3 Member Function Documentation

3.8.3.1 initializeBus()

```
void I2C::initializeBus ( )  [private]
```

I2C bus initialization.

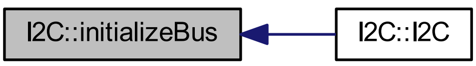
This function initializes the I2C bus, it resets the bus and configure the bitrate as requested. Bitrate is configured according to formula in the ATMEGA2560 datasheet : SCL freq = F_CPU / (16 + 2∗TWBR∗(4^TWPS)). Prescaler value is fixed to 1 (TWPS1 = 0 and TWPS0 = 0), then only TWBR value shall be computed.

Returns

Nothing

Definition at line 76 of file I2C.cpp.

Here is the caller graph for this function:



3.8.3.2 setBitRate()

```
void I2C::setBitRate (
            uint32_t l_bitrate )
```

Variable bitrate setting function.

This function sets the class variable bitrate as requested in parameter.

Parameters

in	*l_bitrate*	Requested bitrate (in Hz)

Returns

Nothing

Definition at line 71 of file I2C.cpp.

3.8.3.3 setTxAddress()

```
void I2C::setTxAddress (
            uint8_t address )
```

Setting function for Tx I2C address.

This function sets the given Tx I2C address in the internal class variable.

Parameters

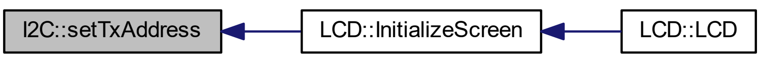
in	*address*	Requested Tx address

Returns

Nothing

Definition at line 66 of file I2C.cpp.

Here is the caller graph for this function:



3.8.3.4 writeByte()

```
bool I2C::writeByte (
            uint8_t * data )
```

Byte sending function.

This function sends one byte on I2C bus

Parameters

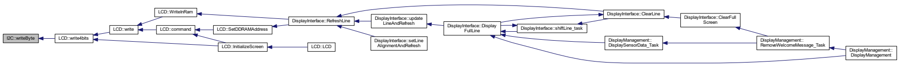
`in`	*data*	Pointer to the data to send

Returns

True if transmission is completed, False if an error has occurred

Definition at line 23 of file I2C.cpp.

Here is the caller graph for this function:



3.8.4 Member Data Documentation

3.8.4.1 bitrate

`uint32_t I2C::bitrate [private]`

Definition at line 63 of file I2C.h.

3.8.4.2 tx_address

`uint8_t I2C::tx_address [private]`

Definition at line 62 of file I2C.h.

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

- I2C.h
- I2C.cpp

3.9 keepAliveLed Class Reference

Class for keep-alive LED blinking.

`#include <keepAliveLed.h>`

Public Member Functions

- keepAliveLed ()
  Class constructor.

Static Public Member Functions

- static void blinkLed_task ()
  Task for LED blinking.

3.9.1 Detailed Description

Class for keep-alive LED blinking.

This class defines all functions to make keep-alive LED blink

Definition at line 22 of file keepAliveLed.h.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 keepAliveLed()

```
keepAliveLed::keepAliveLed ( )
```

Class constructor.

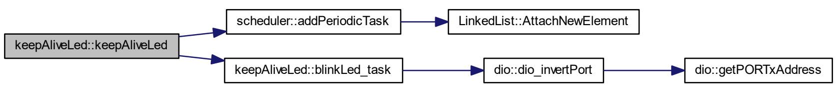
This function initializes the class keppAliveLed

Returns

Nothing

Definition at line 36 of file keepAliveLed.cpp.

Here is the call graph for this function:



3.9.3 Member Function Documentation

3.9.3.1 blinkLed_task()

```
void keepAliveLed::blinkLed_task ( ) [static]
```

Task for LED blinking.

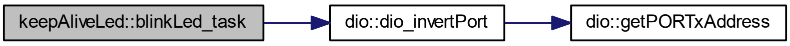
This function is inserted into the scheduler. It changes the state of the LED output to make it blink

Returns

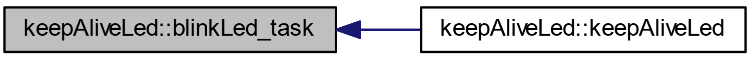
Nothing

Definition at line 42 of file keepAliveLed.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

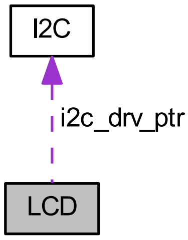
- keepAliveLed.h
- keepAliveLed.cpp

3.10 LCD Class Reference

Class for LCD S2004A display driver.

```
#include <LCD.h>
```

Collaboration diagram for LCD:



Public Member Functions

- LCD (const T_LCD_conf_struct ∗init_conf)

  LCD class constructor.
- void command (T_LCD_command cmd)

  LCD command management function.
- void ConfigureBacklight (bool enable)

  Backlight configuration function.
- void ConfigureLineNumber (bool param)

  Line type configuration function.
- void ConfigureFontType (bool param)

  Font configuration function.
- void ConfigureDisplayOnOff (bool param)

  Display configuration function.
- void ConfigureCursorOnOff (bool param)

  Cursor configuration function.
- void ConfigureCursorBlink (bool param)

  Cursor blinking configuration function.
- void ConfigureEntryModeDir (bool param)

  Entry mode direction configuration function.
- void ConfigureEntryModeShift (bool param)

  Entry mode shift configuration function.
- void ConfigureI2CAddr (uint8_t param)

  I2C address configuration function.
- void SetDDRAMAddress (uint8_t addr)

  DDRAM address setting function.
- uint8_t GetDDRAMAddress ()

  DDRAM address get function.
- void WriteInRam (uint8_t a_char, T_LCD_ram_area area)

  Screen RAM write function.
- bool GetLineNumberCnf ()

  Number of line get function.

Private Member Functions

- void write4bits (uint8_t data)

    I2C write function for 4-bits mode.
- void write (uint8_t data, T_LCD_config_mode mode)

    I2C write function.
- void InitializeScreen ()

    Screen configuration function.

Private Attributes

- bool backlight_enable
- bool cnfLineNumber
- bool cnfFontType
- bool cnfDisplayOnOff
- bool cnfCursorOnOff
- bool cnfCursorBlink
- bool cnfEntryModeDir
- bool cnfEntryModeShift
- uint8_t cnfI2C_addr
- I2C ∗ i2c_drv_ptr
- uint8_t ddram_addr

3.10.1 Detailed Description

Class for LCD S2004A display driver.

This class handles functions managing LCD display S2004a on I2C bus

Definition at line 147 of file LCD.h.

3.10.2 Constructor & Destructor Documentation

3.10.2.1 LCD()

```
LCD::LCD (
            const T_LCD_conf_struct * init_conf )
```

LCD class constructor.

This constructor function initializes the class LCD and calls screen configuration function. It also creates a new instance of the I2C driver if needed.

Parameters

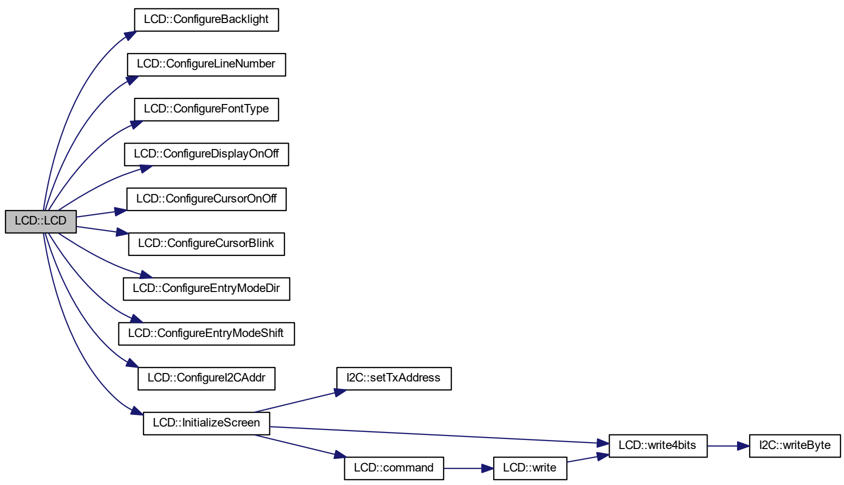
in	*init_conf*	Initial configuration structure

Returns

Nothing

Definition at line 27 of file LCD.cpp.

Here is the call graph for this function:



3.10.3 Member Function Documentation

3.10.3.1 command()

```
void LCD::command (
            T_LCD_command cmd )
```

LCD command management function.

This function sends the requested command to the LCD screen. It builds the 8-bit command word and sends it on I2C bus.

Parameters

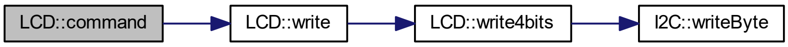
in	*cmd*	Requested command

Returns

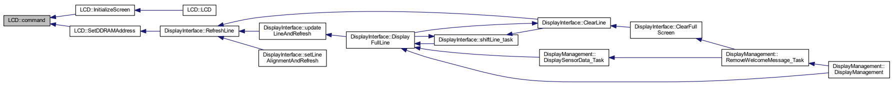
Nothing

Definition at line 134 of file LCD.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.10.3.2 ConfigureBacklight()

```
void LCD::ConfigureBacklight (
            bool enable )  [inline]
```

Backlight configuration function.

This function configures the screen backlight (enable or disable) according to the parameter enable.

Parameters

in	*enable*	True if backlight shall be on, False otherwise

Returns

Nothing

Definition at line 178 of file LCD.h.

Here is the caller graph for this function:

3.10.3.3 ConfigureCursorBlink()

```
void LCD::ConfigureCursorBlink (
            bool param ) [inline]
```

Cursor blinking configuration function.

This function configures the cursor blinking (on or off mode) according to the parameter.

Parameters

in	*param*	Configuration value

Returns

Nothing

Definition at line 238 of file LCD.h.

Here is the caller graph for this function:

3.10.3.4 ConfigureCursorOnOff()

```
void LCD::ConfigureCursorOnOff (
            bool param ) [inline]
```

Cursor configuration function.

This function configures the cursor (on or off mode) according to the parameter.

Parameters

in	*param*	Configuration value

Returns

Nothing

Definition at line 226 of file LCD.h.

Here is the caller graph for this function:

3.10.3.5 ConfigureDisplayOnOff()

```
void LCD::ConfigureDisplayOnOff (
            bool param )  [inline]
```

Display configuration function.

This function configures the display (on or off mode) according to the parameter.

Parameters

in	*param*	Configuration value

Returns

Nothing

Definition at line 214 of file LCD.h.

Here is the caller graph for this function:

3.10.3.6 ConfigureEntryModeDir()

```
void LCD::ConfigureEntryModeDir (
            bool param )  [inline]
```

Entry mode direction configuration function.

This function configures the direction of entry mode (right or left) according to the parameter.

Parameters

in	*param*	Configuration value

Returns

Nothing

Definition at line 250 of file LCD.h.

Here is the caller graph for this function:

3.10.3.7 ConfigureEntryModeShift()

```
void LCD::ConfigureEntryModeShift (
            bool param )  [inline]
```

Entry mode shift configuration function.

This function configures the display shift of entry mode (enable or disable) according to the parameter.

Parameters

in	*param*	Configuration value

Returns

Nothing

Definition at line 262 of file LCD.h.

Here is the caller graph for this function:

3.10.3.8 ConfigureFontType()

```
void LCD::ConfigureFontType (
            bool param ) [inline]
```

Font configuration function.

This function configures the font type of the screen (5∗8 or 5∗11 dots) according to the parameter.

Parameters

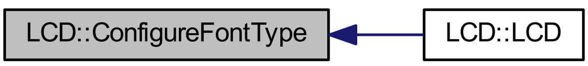
in	*param*	Configuration value

Returns

Nothing

Definition at line 202 of file LCD.h.

Here is the caller graph for this function:



3.10.3.9 ConfigureI2CAddr()

```
void LCD::ConfigureI2CAddr (
            uint8_t param )  [inline]
```

I2C address configuration function.

This function configures the I2V address of the LCD screen according to the parameter.

Parameters

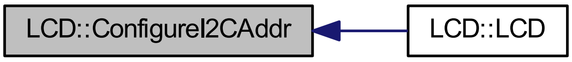
in	*param*	I2C address

Returns

Nothing

Definition at line 274 of file LCD.h.

Here is the caller graph for this function:



3.10.3.10 ConfigureLineNumber()

```
void LCD::ConfigureLineNumber (
            bool param )  [inline]
```

Line type configuration function.

This function configures the line number configuration of the screen (1 or 2 lines mode) according to the parameter.

Parameters

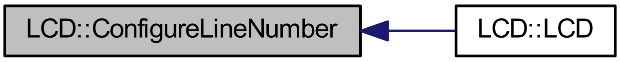
in	*param*	Configuration value

Returns

Nothing

Definition at line 190 of file LCD.h.

Here is the caller graph for this function:



3.10.3.11 GetDDRAMAddress()

`uint8_t LCD::GetDDRAMAddress ( ) [inline]`

DDRAM address get function.

This function return the value of the current DDRAM address stored in internal variable ddram_addr.

Returns

Current DDRAM address

Definition at line 294 of file LCD.h.

3.10.3.12 GetLineNumberCnf()

`bool LCD::GetLineNumberCnf ( ) [inline]`

Number of line get function.

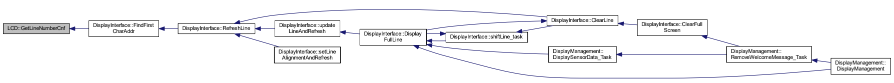
This function returns the line number configuration of the screen : 1 or 2 lines mode.

Returns

Line number configuration

Definition at line 316 of file LCD.h.

Here is the caller graph for this function:



3.10.3.13 InitializeScreen()

```
void LCD::InitializeScreen ( )  [private]
```

Screen configuration function.

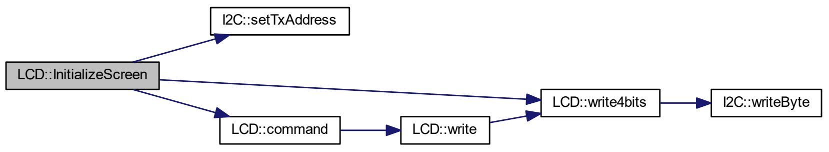
This function configures the LCD screen. It's must be called during initialization phase, or the screen won't be usable. The configuration process is described in LCD datasheet J2004A-GFDN-DYNC

Returns

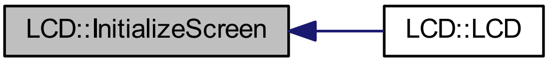
Nothing

Definition at line 82 of file LCD.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.10.3.14 SetDDRAMAddress()

```
void LCD::SetDDRAMAddress (
            uint8_t addr )
```

DDRAM address setting function.

This function updates the DDRAM address according to the given parameter. The parameter is checked against limits to be sure the address stays always coherent. It also calls the command function to update screen accordingly.

Parameters

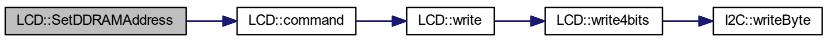
in	*addr*	New DDRAM address

Returns

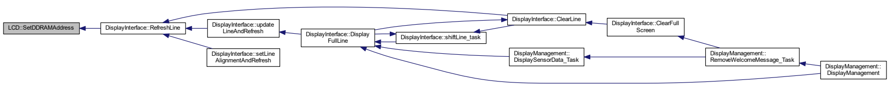
Nothing

Definition at line 177 of file LCD.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.10.3.15 write()

```
void LCD::write (
            uint8_t data,
            T_LCD_config_mode mode )  [private]
```

I2C write function.

This function writes the requested data on I2C bus. It's assumed we only perform write operation so the R/W bit is forced LOW. It's also assumed we work in 4-bit mode, then two calls of write4bits are performed, first with bits 4-7 of data, second with bits 0-3.

Parameters

in	*data*	8-bit data for D0-7 pins of screen
in	*mode*	Requested mode for LCD communication

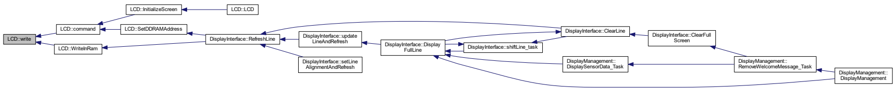
Returns

Nothing

Definition at line 71 of file LCD.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



3.10.3.16 write4bits()

```
void LCD::write4bits (
            uint8_t data )  [private]
```

I2C write function for 4-bits mode.

This function sends the requested 8-bits data on the I2C bus. The backlight pin is also set/clear according to the configuration. The function sends the data a first time with EN pin set, then a second time with EN bit clear.

Parameters

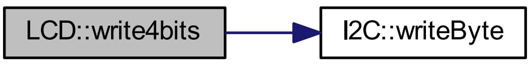
in	*data*	8-bit data to send

Returns

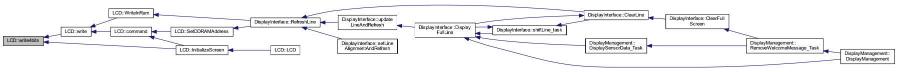
Nothing

Definition at line 54 of file LCD.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.10.3.17 WriteInRam()

```
void LCD::WriteInRam (
            uint8_t a_char,
            T_LCD_ram_area area )
```

Screen RAM write function.

This function writes in the memorized RAM address the character given as parameter. After a write the screen automatically increment/decrement the RAM address, so we do the same in the function to stay coherent. Currently only DDRAM write is implemented.

Parameters

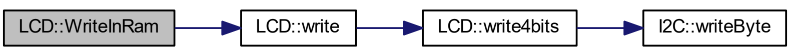
in	*a_char*	Data byte to write in RAM
in	*area*	Area in RAM where the data will be written : DDRAM or CGRAM

Returns

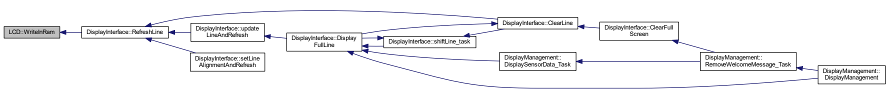
Nothing

Definition at line 199 of file LCD.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.10.4 Member Data Documentation

3.10.4.1 backlight_enable

```
bool LCD::backlight_enable  [private]
```

Backlight enable flag

Definition at line 324 of file LCD.h.

3.10.4.2 cnfCursorBlink

```
bool LCD::cnfCursorBlink  [private]
```

Cursor blinking configuration : 1 = cursor blink is on, 0 = cursor blink is off

Definition at line 329 of file LCD.h.

3.10.4.3 cnfCursorOnOff

```
bool LCD::cnfCursorOnOff  [private]
```

Cursor configuration : 1 = cursor on, 0 = cursor off

Definition at line 328 of file LCD.h.

3.10.4.4 cnfDisplayOnOff

```
bool LCD::cnfDisplayOnOff  [private]
```

Display configuration : 1 = display on, 0 = display off

Definition at line 327 of file LCD.h.

3.10.4.5 cnfEntryModeDir

```
bool LCD::cnfEntryModeDir  [private]
```

Entry mode direction configuration : 1 = cursor moves to right when DDRAM address is incremented, 0 = cursor moves to left when DDRAM address is incremented

Definition at line 330 of file LCD.h.

3.10.4.6 cnfEntryModeShift

```
bool LCD::cnfEntryModeShift [private]
```

Entry mode configuration : 0 = no display shift is performed after a DDRAM read, 1 = a shift is performed

Definition at line 331 of file LCD.h.

3.10.4.7 cnfFontType

```
bool LCD::cnfFontType [private]
```

Font type configuration, 0 = 5∗8 dots, 1 = 5∗11 dots

Definition at line 326 of file LCD.h.

3.10.4.8 cnfI2C_addr

```
uint8_t LCD::cnfI2C_addr [private]
```

I2C address of the LCD screen

Definition at line 332 of file LCD.h.

3.10.4.9 cnfLineNumber

```
bool LCD::cnfLineNumber [private]
```

Display line number configuration, 0 = 1-line mode, 1 = 2-line mode

Definition at line 325 of file LCD.h.

3.10.4.10 ddram_addr

```
uint8_t LCD::ddram_addr [private]
```

Screen DDRAM address

Definition at line 336 of file LCD.h.

3.10.4.11 i2c_drv_ptr

`I2C* LCD::i2c_drv_ptr [private]`

Pointer to the I2C driver object

Definition at line 334 of file LCD.h.

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

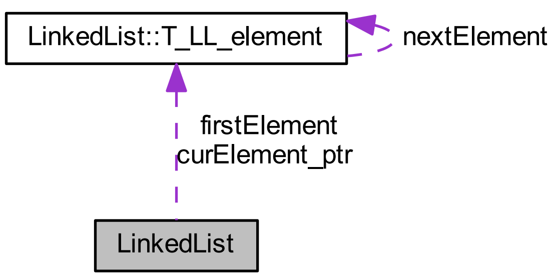
- LCD.h
- LCD.cpp

3.11 LinkedList Class Reference

Linked list class.

`#include <LinkedList.h>`

Collaboration diagram for LinkedList:



Classes

- struct T_LL_element

  Type defining a linked list element.

Public Member Functions

- LinkedList ()

  Class constructor.
- ∼LinkedList ()

  Class destructor.
- void AttachNewElement (void ∗data_ptr)

  Add an new element to the list.
- bool RemoveElement (CompareFctPtr_t comparisonFct_ptr, void ∗reference_ptr)

  Removes an element from the chain.
- void ∗ getCurrentElement ()

  Current element get function.
- bool MoveToNextElement ()

  Move to next element function.
- void ResetElementPtr ()

  Resets element pointer.
- bool IsLLEmpty ()

  Empty linked list.
- bool FindElement (CompareFctPtr_t comparisonFct_ptr, void ∗reference_ptr, void ∗∗chainElement_ptr)

  Element finding function.

Private Types

- typedef struct LinkedList::T_LL_element T_LL_element

  Type defining a linked list element.

Private Attributes

- T_LL_element ∗ firstElement
- T_LL_element ∗ curElement_ptr

3.11.1 Detailed Description

Linked list class.

This class defines a linked list and the associated services.
All classes using a linked list with this interface shall implement a comparison function used to find the list element to remove. This function shall have the following prototype : static bool LLElementCompare(void∗ LLElement, void∗ CompareElement);

Definition at line 22 of file LinkedList.h.

3.11.2 Member Typedef Documentation

3.11.2.1 T_LL_element

`typedef struct LinkedList::T_LL_element LinkedList::T_LL_element [private]`

Type defining a linked list element.

This structure defines a linked list element. An element is defined by a pointer to the attached data and a pointer to the next element.

3.11.3 Constructor & Destructor Documentation

3.11.3.1 LinkedList()

`LinkedList::LinkedList ( )`

Class constructor.

This constructor initializes a linked list

Returns

Nothing

Definition at line 37 of file LinkedList.cpp.

3.11.3.2 ∼LinkedList()

`LinkedList::∼LinkedList ( )`

Class destructor.

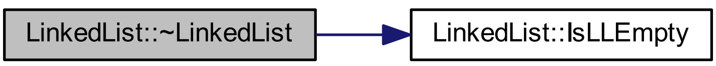
This function deletes the linked list

Returns

Nothing

Definition at line 43 of file LinkedList.cpp.

Here is the call graph for this function:



3.11.4 Member Function Documentation

3.11.4.1 AttachNewElement()

```
void LinkedList::AttachNewElement (
            void * data_ptr )
```

Add an new element to the list.

This function adds a new element at the end of the list. The data pointer to attach to the element is given in parameter

Parameters

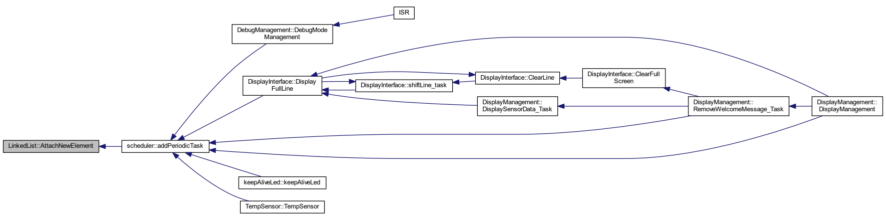
in	*data_ptr*	Pointer to the data element

Returns

Nothing

Definition at line 61 of file LinkedList.cpp.

Here is the caller graph for this function:



3.11.4.2 FindElement()

```
bool LinkedList::FindElement (
            CompareFctPtr_t comparisonFct_ptr,
            void * reference_ptr,
            void ** chainElement_ptr )
```

Element finding function.

This function finds the given element reference_ptr inside the chain. The comparison between the elements of the chain and the reference element is done using the given comparison function.

Parameters

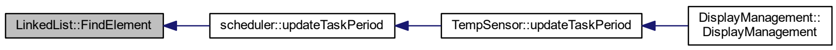
`in`	*comparisonFct_ptr*	Pointer to the comparison function
`in`	*reference_ptr*	Pointer to the element to find in the chain
`out`	*chainElement_ptr*	Pointer to pointer to the found element

Returns

True if the element has been found in the chain, false otherwise

Definition at line 135 of file LinkedList.cpp.

Here is the caller graph for this function:



3.11.4.3 getCurrentElement()

`void* LinkedList::getCurrentElement ( )  [inline]`

Current element get function.

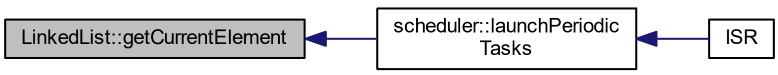
This function returns a pointer to the current pointed data in the chain.

Returns

Pointer to the current data

Definition at line 67 of file LinkedList.h.

Here is the caller graph for this function:



3.11.4.4 IsLLEmpty()

```
bool LinkedList::IsLLEmpty ( )
```

Empty linked list.

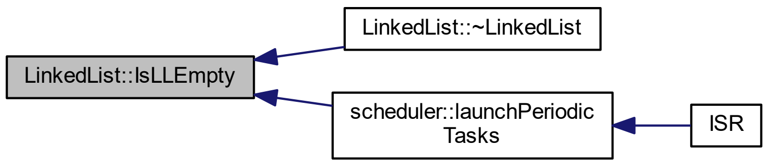
This function checks whether the linked list is empty or not (pointer to first element is equal to 0 or not).

Returns

True if the list is empty, false otherwise

Definition at line 127 of file LinkedList.cpp.

Here is the caller graph for this function:



3.11.4.5 MoveToNextElement()

```
bool LinkedList::MoveToNextElement ( )
```

Move to next element function.

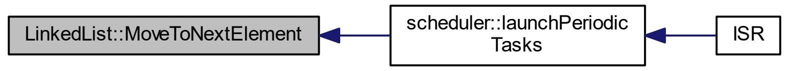
This function moves the element pointer to the next element of the chain.

Returns

True if the next element exists, false if there is no next element

Definition at line 113 of file LinkedList.cpp.

Here is the caller graph for this function:



3.11.4.6 RemoveElement()

```
bool LinkedList::RemoveElement (
            CompareFctPtr_t comparisonFct_ptr,
            void * reference_ptr )
```

Removes an element from the chain.

This function removes an element from the chain. To know which element shall be removed, we use the comparison function given in parameter. This function is called with two parameters : the data pointer from the chain and the reference pointer given as parameter.

Parameters

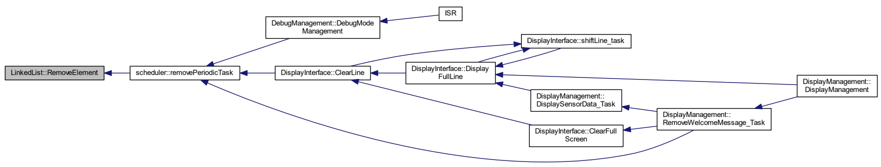
in	*comparisonFct_ptr*	Pointer to the comparison function to use
in	*reference_ptr*	Pointer to the reference data used for comparison

Returns

True if the element has been correctly removed from the chain, false otherwise

Definition at line 86 of file LinkedList.cpp.

Here is the caller graph for this function:



3.11.4.7 ResetElementPtr()

```
void LinkedList::ResetElementPtr ( )  [inline]
```

Resets element pointer.

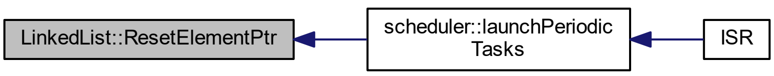
This function sets the element pointer to the first element of the chain.

Returns

Nothing

Definition at line 86 of file LinkedList.h.

Here is the caller graph for this function:



3.11.5 Member Data Documentation

3.11.5.1 curElement_ptr

`T_LL_element* LinkedList::curElement_ptr [private]`

Pointer to the current element of the list

Definition at line 125 of file LinkedList.h.

3.11.5.2 firstElement

`T_LL_element* LinkedList::firstElement [private]`

Pointer to the first element of the list

Definition at line 124 of file LinkedList.h.

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

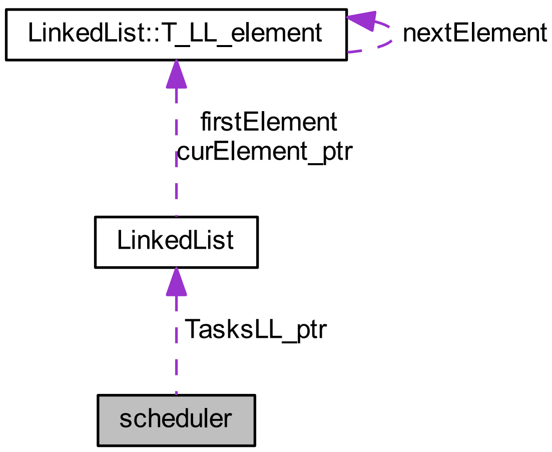
- LinkedList.h
- LinkedList.cpp

3.12 scheduler Class Reference

Scheduler class.

`#include <scheduler.h>`

Collaboration diagram for scheduler:



Classes

- struct Task_t

    Type defining a task structure.

Public Member Functions

- scheduler ()

    scheduler class constructor
- void launchPeriodicTasks ()

    Main scheduler function.
- void startScheduling ()

    Starts the tasks scheduling.
- void addPeriodicTask (TaskPtr_t task_ptr, uint16_t a_period)

    Add a task into the scheduler.
- bool removePeriodicTask (TaskPtr_t task_ptr)

    Remove a task from the scheduler.
- uint32_t getPitNumber ()

    Get function for PIT number.
- bool updateTaskPeriod (TaskPtr_t task_ptr, uint16_t period)

    Task period update function.

Static Public Member Functions

- static bool LLElementCompare (void ∗LLElement, void ∗CompareElement)

    Linked list comparison function.

Private Types

- typedef struct scheduler::Task_t Task_t

    Type defining a task structure.

Private Attributes

- LinkedList ∗ TasksLL_ptr
- uint32_t pit_number

3.12.1 Detailed Description

Scheduler class.

This class defines the scheduler of the system.
It is called by the main interrupt and calls successively all applicative functions according to their recurrence time.
All tasks called by the scheduler shall have the following prototype : static void task();

Definition at line 30 of file scheduler.h.

3.12.2 Member Typedef Documentation

3.12.2.1 Task_t

```
typedef struct scheduler::Task_t scheduler::Task_t  [private]
```

Type defining a task structure.

This structure defines a task. A task is defined by a function to call (defined by its pointer) and a period.

3.12.3 Constructor & Destructor Documentation

3.12.3.1 scheduler()

```
scheduler::scheduler ( )
```

scheduler class constructor

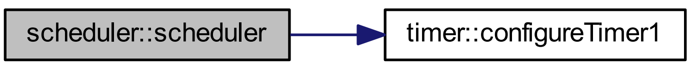
This function initializes the class scheduler

Returns

Nothing

Definition at line 40 of file scheduler.cpp.

Here is the call graph for this function:



3.12.4 Member Function Documentation

3.12.4.1 addPeriodicTask()

```
void scheduler::addPeriodicTask (
            TaskPtr_t task_ptr,
            uint16_t a_period )
```

Add a task into the scheduler.

This function create a new task in the scheduler linked to the function task_ptr with a period a_period and an ID a_task_id

Parameters

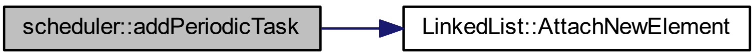
in	*task_ptr*	Pointer to the task which will be added
in	*a_period*	Period of the new task

Returns

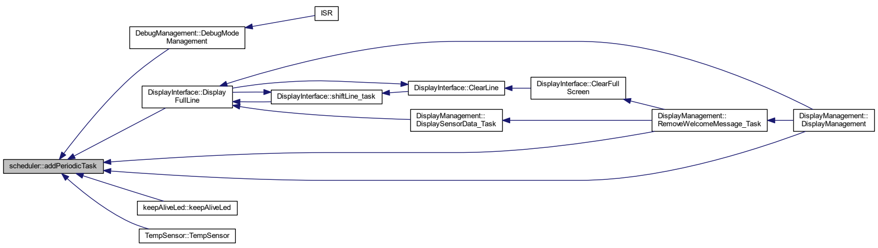
Nothing

Definition at line 93 of file scheduler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.12.4.2 getPitNumber()

```
uint32_t scheduler::getPitNumber ( )
```

Get function for PIT number.

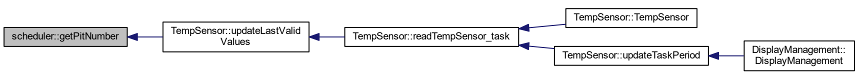
This function returns the PIT number

Returns

PIT number

Definition at line 105 of file scheduler.cpp.

Here is the caller graph for this function:



3.12.4.3 launchPeriodicTasks()

```
void scheduler::launchPeriodicTasks ( )
```

Main scheduler function.

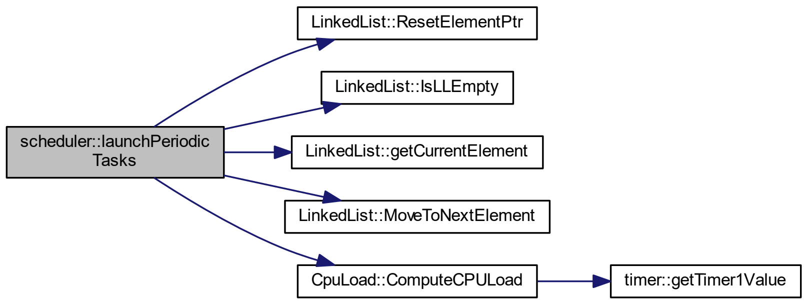
This function launches the scheduled tasks according to current software time and task configuration

Returns

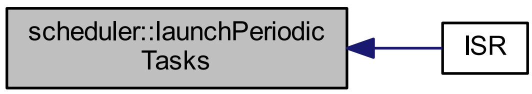
Nothing

Definition at line 52 of file scheduler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.12.4.4 LLElementCompare()

```
bool scheduler::LLElementCompare (
            void * LLElement,
            void * CompareElement ) [static]
```

Linked list comparison function.

This function is called by the linked list class to compare one element of the list to a given element. In the class scheduler, the LLElement is a task pointer (containing a function pointer and a period), and the compareElement a function pointer. The comparison will be done between the two function pointer.

Parameters

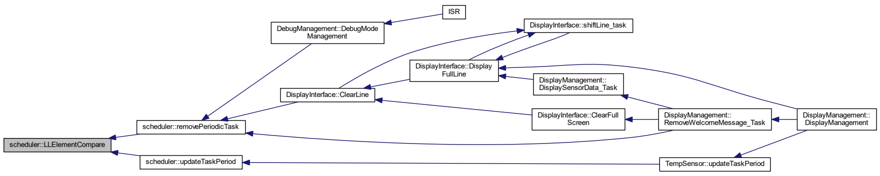
in	*LLElement*	Pointer to the linked list element
in	*CompareElement*	Pointer to the element to the compare

Returns

True if both elements are identical, false otherwise

Definition at line 116 of file scheduler.cpp.

Here is the caller graph for this function:



3.12.4.5 removePeriodicTask()

```
bool scheduler::removePeriodicTask (
            TaskPtr_t task_ptr )
```

Remove a task from the scheduler.

This function finds the task defined by task_ptr in the scheduler and removes it.

Parameters

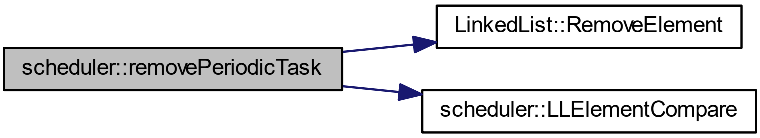
in	*task_ptr*	address of the task to remove from scheduler

Returns

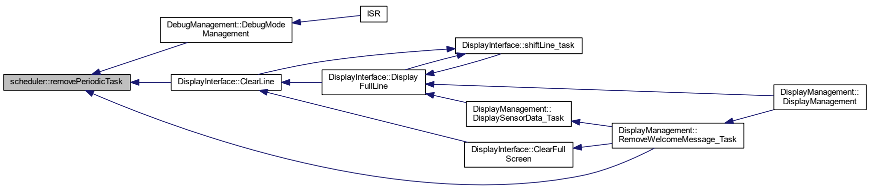
TRUE if the task has been removed, FALSE if the task does not exist in the scheduler

Definition at line 111 of file scheduler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.12.4.6 startScheduling()

```
void scheduler::startScheduling ( )
```

Starts the tasks scheduling.

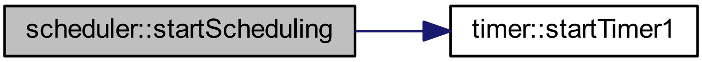
This function starts the timer which will trigger an interrupt every software period. When the interrupt is raised the scheduler will launch applications

Returns

Nothing

Definition at line 87 of file scheduler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

3.12.4.7 updateTaskPeriod()

```
bool scheduler::updateTaskPeriod (
            TaskPtr_t task_ptr,
            uint16_t period )
```

Task period update function.

This function updates the period of the given task. The task is never stopped during the process, only the period value is updated.

Parameters

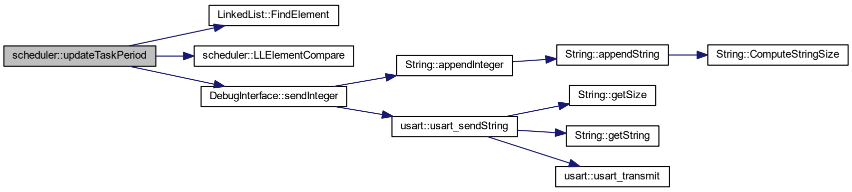
in	*task_ptr*	Pointer of the task to update
in	*period*	New period of the task

Returns

True if the update has been correctly done, false otherwise

Definition at line 127 of file scheduler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

3.12.5 Member Data Documentation

3.12.5.1 pit_number

`uint32_t scheduler::pit_number [private]`

Counter of periodic interrupts

Definition at line 127 of file scheduler.h.

3.12.5.2 TasksLL_ptr

`LinkedList* scheduler::TasksLL_ptr [private]`

Pointer to the linked list object containing the tasks

Definition at line 125 of file scheduler.h.

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

- scheduler.h
- scheduler.cpp

3.13 String Class Reference

String management class.

```
#include <String.h>
```

Public Member Functions

- String (const uint8_t *str)

    Class constructor.
- String ()

    Class constructor.
- ∼String ()

    Class destructor.
- uint8_t * getString ()

    String pointer get function.
- uint8_t getSize ()

    Size get function.
- void appendString (uint8_t *str)

    String adding function.
- void appendInteger (uint16_t value, uint8_t base)

    Integer adding function.
- void appendBool (bool data, bool isText)

    Boolean adding function.
- void appendChar (uint8_t data)

    Character adding function.
- void Clear ()

    String clear function.

Private Member Functions

- uint8_t ComputeStringSize (uint8_t *str)

    String size computation function.

Private Attributes

- uint8_t * string
- uint8_t size

3.13.1 Detailed Description

String management class.

This class defines string object. It implements some functions to manage chains of characters. The string is limited to 255 characters. It must finish by the character '\0'.

Definition at line 18 of file String.h.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 String() [1/2]

```
String::String (
            const uint8_t * str )
```

Class constructor.

This function initializes the class. The string is initialized with the data given in parameter.

Parameters

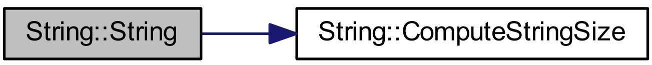
in	*str*	Pointer to initialization string

Returns

Nothing

Definition at line 15 of file String.cpp.

Here is the call graph for this function:



3.13.2.2 String() [2/2]

```
String::String ( )
```

Class constructor.

This function initializes the class with an empty string. The size is set to 0.

Returns

Nothing

Definition at line 33 of file String.cpp.

3.13.2.3 ∼String()

```
String::∼String ( )
```

Class destructor.

This function frees the memory used to contain the string when the object is deleted

Returns

Nothing

Definition at line 39 of file String.cpp.

Here is the call graph for this function:

3.13.3 Member Function Documentation

3.13.3.1 appendBool()

```
void String::appendBool (
            bool data,
            bool isText )
```

Boolean adding function.

This functions adds the given boolean data at the end of the main string. The string size is updated accordingly. According to the input parameter isText, the boolean parameter is converted into a string (true/false) or an integer (0/1).

Parameters

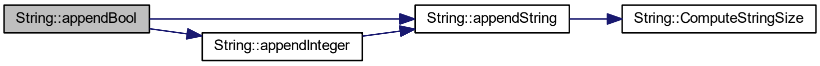
in	*data*	Boolean data to add
in	*isText*	Defines the conversion mode : text or integer

Returns

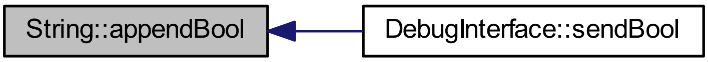
Nothing

Definition at line 121 of file String.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.13.3.2 appendChar()

```
void String::appendChar (
            uint8_t data )
```

Character adding function.

This functions adds the given character at the end of the main string. The string size is updated by 1.

Parameters

in	*data*	1-byte character to add

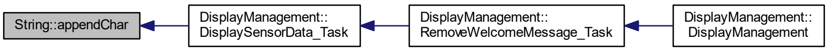
Returns

Nothing

Definition at line 139 of file String.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



3.13.3.3 appendInteger()

```
void String::appendInteger (
            uint16_t value,
            uint8_t base )
```

Integer adding function.

This functions adds the given integer at the end of the main string. The string size is updated accordingly. The integer parameter is first converted into a chain of character according to the base and then added to the string.

Parameters

in	*value*	Integer to add
in	*base*	Base of computation of the integer (between 2 and 36)

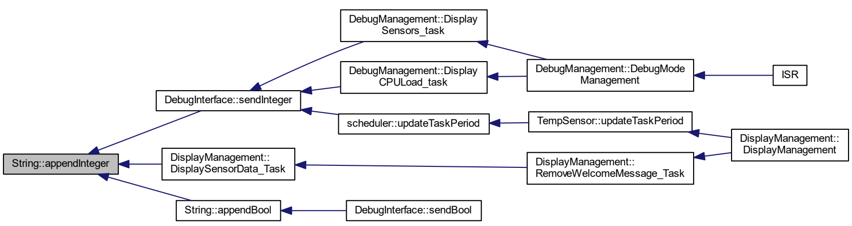
Returns

Nothing

Definition at line 95 of file String.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



3.13.3.4 appendString()

```
void String::appendString (
            uint8_t * str )
```

String adding function.

This functions adds the given string at the end of the main string. The string size is updated accordingly.

Parameters

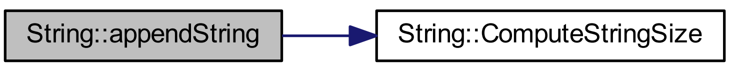
in	*str*	New string to add

Returns

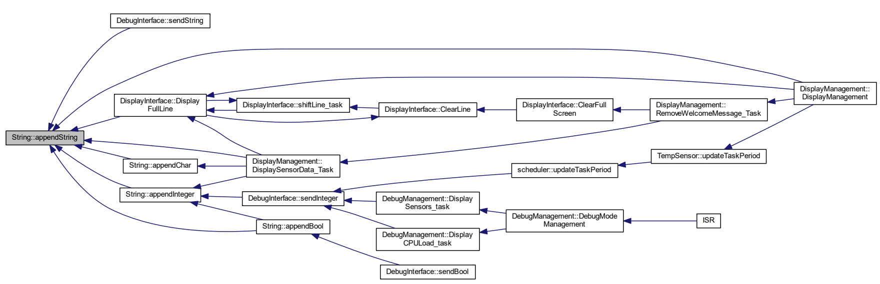
Nothing

Definition at line 57 of file String.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.13.3.5 Clear()

```
void String::Clear ( )
```

String clear function.

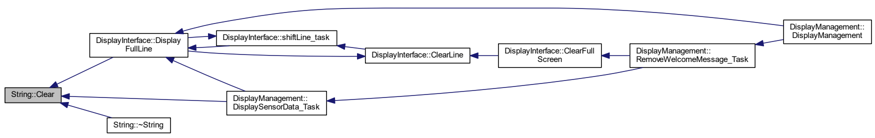
This function clears the string. Size is set to 0 and the memory is freed.

Returns

Nothing

Definition at line 113 of file String.cpp.

Here is the caller graph for this function:



3.13.3.6 ComputeStringSize()

```
uint8_t String::ComputeStringSize (
            uint8_t * str )  [private]
```

String size computation function.

This function computes the sizes of the given string. It counts the number of character between the start of the string given in parameter and the next \0 character.

Parameters

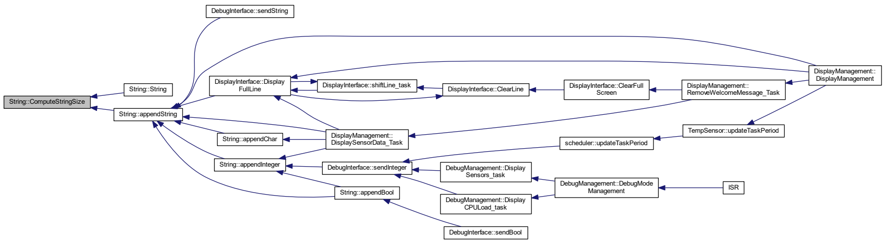
in	*str*	Pointer to the beginning of the string

Returns

Number of character of the string (the \0 is excluded)

Definition at line 44 of file String.cpp.

Here is the caller graph for this function:



3.13.3.7 getSize()

```
uint8_t String::getSize ( ) [inline]
```

Size get function.

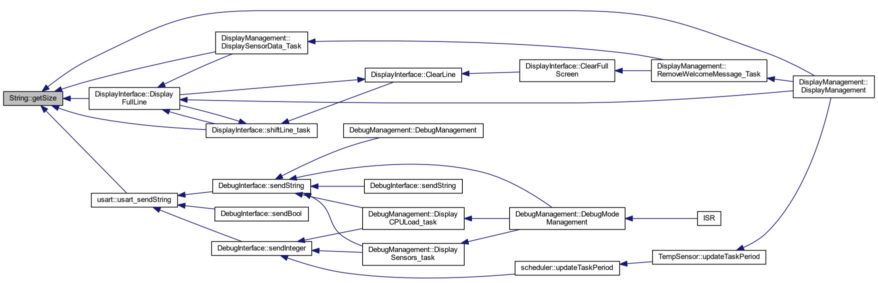
This function returns the size of the string.

Returns

Size of the string

Definition at line 64 of file String.h.

Here is the caller graph for this function:



3.13.3.8 getString()

```
uint8_t* String::getString ( ) [inline]
```

String pointer get function.

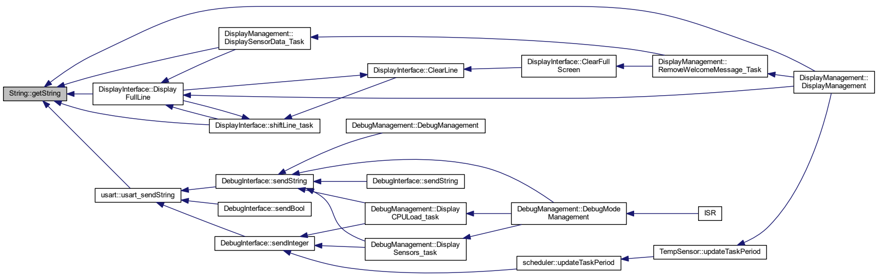
This function returns the pointer to the beginning of the string.

Returns

String pointer

Definition at line 53 of file String.h.

Here is the caller graph for this function:



3.13.4 Member Data Documentation

3.13.4.1 size

```
uint8_t String::size [private]
```

Size of the string (the '\0' at the end of the string is not taken into account

Definition at line 121 of file String.h.

3.13.4.2 string

```
uint8_t* String::string [private]
```

Pointer to the start of the string

Definition at line 120 of file String.h.

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

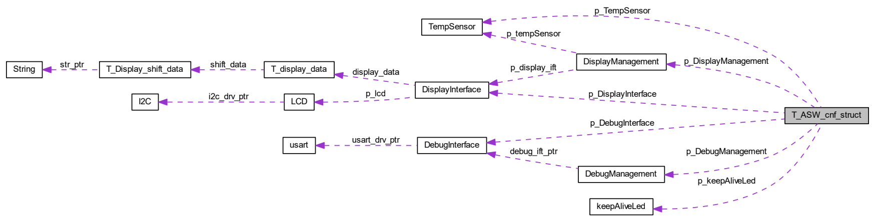
- String.h
- String.cpp

3.14 T_ASW_cnf_struct Struct Reference

ASW configuration structure.

`#include <asw.h>`

Collaboration diagram for T_ASW_cnf_struct:



Public Attributes

- DebugInterface ∗ p_DebugInterface
- keepAliveLed ∗ p_keepAliveLed
- TempSensor ∗ p_TempSensor
- DisplayInterface ∗ p_DisplayInterface
- DisplayManagement ∗ p_DisplayManagement
- DebugManagement ∗ p_DebugManagement

3.14.1 Detailed Description

ASW configuration structure.

This structure contains all pointers to instanced applicative objects

Definition at line 18 of file asw.h.

3.14.2 Member Data Documentation

3.14.2.1 p_DebugInterface

`DebugInterface* T_ASW_cnf_struct::p_DebugInterface`

Pointer to USART debug interface object

Definition at line 20 of file asw.h.

3.14.2.2 p_DebugManagement

```
DebugManagement* T_ASW_cnf_struct::p_DebugManagement
```

Pointer to the DebugManagement object

Definition at line 25 of file asw.h.

3.14.2.3 p_DisplayInterface

```
DisplayInterface* T_ASW_cnf_struct::p_DisplayInterface
```

Pointer to DisplayInterface object

Definition at line 23 of file asw.h.

3.14.2.4 p_DisplayManagement

```
DisplayManagement* T_ASW_cnf_struct::p_DisplayManagement
```

Pointer to DisplayManagement object

Definition at line 24 of file asw.h.

3.14.2.5 p_keepAliveLed

```
keepAliveLed* T_ASW_cnf_struct::p_keepAliveLed
```

Pointer to keepAliveLed object

Definition at line 21 of file asw.h.

3.14.2.6 p_TempSensor

```
TempSensor* T_ASW_cnf_struct::p_TempSensor
```

Pointer to TempSensor object

Definition at line 22 of file asw.h.

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

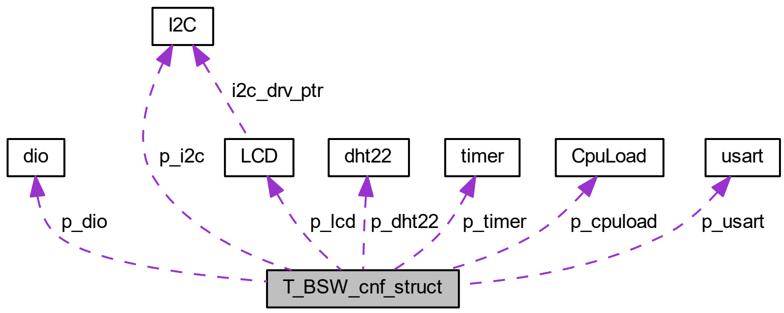
- asw.h

3.15 T_BSW_cnf_struct Struct Reference

BSW configuration structure.

`#include <bsw.h>`

Collaboration diagram for T_BSW_cnf_struct:



Public Attributes

- usart ∗ p_usart
- dio ∗ p_dio
- timer ∗ p_timer
- dht22 ∗ p_dht22
- CpuLoad ∗ p_cpuload
- I2C ∗ p_i2c
- LCD ∗ p_lcd

3.15.1 Detailed Description

BSW configuration structure.

This structure contains all pointers to instanced drivers objects

Definition at line 17 of file bsw.h.

3.15.2 Member Data Documentation

3.15.2.1 p_cpuload

`CpuLoad* T_BSW_cnf_struct::p_cpuload`

Pointer to cpu load library object

Definition at line 23 of file bsw.h.

3.15.2.2 p_dht22

```
dht22* T_BSW_cnf_struct::p_dht22
```

Pointer to dht22 driver object

Definition at line 22 of file bsw.h.

3.15.2.3 p_dio

```
dio* T_BSW_cnf_struct::p_dio
```

Pointer to dio driver object

Definition at line 20 of file bsw.h.

3.15.2.4 p_i2c

```
I2C* T_BSW_cnf_struct::p_i2c
```

Pointer to I2C driver object

Definition at line 24 of file bsw.h.

3.15.2.5 p_lcd

```
LCD* T_BSW_cnf_struct::p_lcd
```

Pointer to LCD driver object

Definition at line 25 of file bsw.h.

3.15.2.6 p_timer

```
timer* T_BSW_cnf_struct::p_timer
```

Pointer to timer driver object

Definition at line 21 of file bsw.h.

3.15.2.7 p_usart

```
usart* T_BSW_cnf_struct::p_usart
```

Pointer to usart driver object

Definition at line 19 of file bsw.h.

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

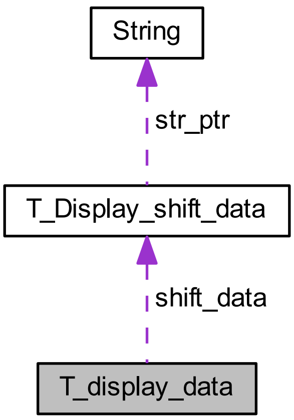
- bsw.h

3.16 T_display_data Struct Reference

Structure containing display data.

```
#include <DisplayInterface.h>
```

Collaboration diagram for T_display_data:



Public Attributes

- bool isEmpty
- T_DisplayInterface_LineDisplayMode mode
- T_DisplayInterface_LineAlignment alignment
- T_Display_shift_data shift_data
- uint8_t display_str [LCD_SIZE_NB_CHAR_PER_LINE]

3.16.1 Detailed Description

Structure containing display data.

This structure contains all data used for screen display

Definition at line 57 of file DisplayInterface.h.

3.16.2 Member Data Documentation

3.16.2.1 alignment

`T_DisplayInterface_LineAlignment T_display_data::alignment`

Line alignment

Definition at line 61 of file DisplayInterface.h.

3.16.2.2 display_str

`uint8_t T_display_data::display_str[LCD_SIZE_NB_CHAR_PER_LINE]`

Current string displayed on the screen

Definition at line 63 of file DisplayInterface.h.

3.16.2.3 isEmpty

`bool T_display_data::isEmpty`

Flag indicating if the line is empty or not

Definition at line 59 of file DisplayInterface.h.

3.16.2.4 mode

`T_DisplayInterface_LineDisplayMode T_display_data::mode`

Current display mode

Definition at line 60 of file DisplayInterface.h.

3.16.2.5 shift_data

`T_Display_shift_data T_display_data::shift_data`

Shift data for the current line

Definition at line 62 of file DisplayInterface.h.

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

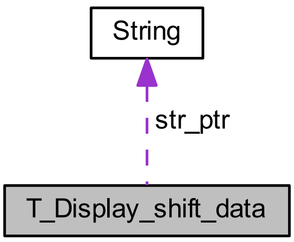
- DisplayInterface.h

3.17 T_Display_shift_data Struct Reference

Structure containing shift data.

`#include <DisplayInterface.h>`

Collaboration diagram for T_Display_shift_data:



Public Attributes

- String ∗ str_ptr
- uint8_t ∗ str_cur_ptr
- uint8_t temporization

3.17.1 Detailed Description

Structure containing shift data.

This structure contains all useful data for line shifting. These data need to be kept between each call of the periodic function.

Definition at line 45 of file DisplayInterface.h.

3.17.2 Member Data Documentation

3.17.2.1 str_cur_ptr

```
uint8_t* T_Display_shift_data::str_cur_ptr
```

Pointer to the address of the first displayed character

Definition at line 48 of file DisplayInterface.h.

3.17.2.2 str_ptr

```
String* T_Display_shift_data::str_ptr
```

Pointer to the start address of the string

Definition at line 47 of file DisplayInterface.h.

3.17.2.3 temporization

```
uint8_t T_Display_shift_data::temporization
```

Shifting period

Definition at line 49 of file DisplayInterface.h.

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

- DisplayInterface.h

3.18 T_LCD_conf_struct Struct Reference

Structure defining LCD configuration.

```
#include <LCD.h>
```

Public Attributes

- uint32_t i2c_bitrate
- uint8_t i2c_addr
- bool backlight_en
- bool lineNumber_cnf
- bool fontType_cnf
- bool display_en
- bool cursor_en
- bool cursorBlink_en
- bool entryModeDir
- bool entryModeShift

3.18.1 Detailed Description

Structure defining LCD configuration.

Definition at line 128 of file LCD.h.

3.18.2 Member Data Documentation

3.18.2.1 backlight_en

```
bool T_LCD_conf_struct::backlight_en
```

Screen backlight enable flag

Definition at line 132 of file LCD.h.

3.18.2.2 cursor_en

```
bool T_LCD_conf_struct::cursor_en
```

Screen cursor enable flag

Definition at line 136 of file LCD.h.

3.18.2.3 cursorBlink_en

```
bool T_LCD_conf_struct::cursorBlink_en
```

Screen cursor blinking enable flag

Definition at line 137 of file LCD.h.

3.18.2.4 display_en

```
bool T_LCD_conf_struct::display_en
```

Screen display enable flag

Definition at line 135 of file LCD.h.

3.18.2.5 entryModeDir

```
bool T_LCD_conf_struct::entryModeDir
```

Entry mode direction configuration

Definition at line 138 of file LCD.h.

3.18.2.6 entryModeShift

```
bool T_LCD_conf_struct::entryModeShift
```

Entry mode shift configuration

Definition at line 139 of file LCD.h.

3.18.2.7 fontType_cnf

```
bool T_LCD_conf_struct::fontType_cnf
```

Font configuration

Definition at line 134 of file LCD.h.

3.18.2.8 i2c_addr

```
uint8_t T_LCD_conf_struct::i2c_addr
```

I2C address if the screen

Definition at line 131 of file LCD.h.

3.18.2.9 i2c_bitrate

`uint32_t T_LCD_conf_struct::i2c_bitrate`

I2C bitrate needed by the LCD screen

Definition at line 130 of file LCD.h.

3.18.2.10 lineNumber_cnf

`bool T_LCD_conf_struct::lineNumber_cnf`

Screen line number configuration (1 or 2 lines)

Definition at line 133 of file LCD.h.

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

- LCD.h

3.19 LinkedList::T_LL_element Struct Reference

Type defining a linked list element.

Collaboration diagram for LinkedList::T_LL_element:

Public Attributes

- void ∗ data_ptr
- T_LL_element ∗ nextElement

3.19.1 Detailed Description

Type defining a linked list element.

This structure defines a linked list element. An element is defined by a pointer to the attached data and a pointer to the next element.

Definition at line 117 of file LinkedList.h.

3.19.2 Member Data Documentation

3.19.2.1 data_ptr

```
void* LinkedList::T_LL_element::data_ptr
```

Definition at line 119 of file LinkedList.h.

3.19.2.2 nextElement

```
T_LL_element* LinkedList::T_LL_element::nextElement
```

Definition at line 120 of file LinkedList.h.

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

- LinkedList.h

3.20 scheduler::Task_t Struct Reference

Type defining a task structure.

Public Attributes

- TaskPtr_t TaskPtr
- uint16_t period

3.20.1 Detailed Description

Type defining a task structure.

This structure defines a task. A task is defined by a function to call (defined by its pointer) and a period.

Definition at line 118 of file scheduler.h.

3.20.2 Member Data Documentation

3.20.2.1 period

```
uint16_t scheduler::Task_t::period
```

Period of the task

Definition at line 121 of file scheduler.h.

3.20.2.2 TaskPtr

```
TaskPtr_t scheduler::Task_t::TaskPtr
```

Pointer to the task

Definition at line 120 of file scheduler.h.

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

- scheduler.h

3.21 TempSensor Class Reference

Class for temperature sensor.

```
#include <TempSensor.h>
```

Public Member Functions

- TempSensor ()
  Class constructor.
- uint16_t ∗ getTempPtr ()
  Get pointer to data raw_temperature.
- uint16_t ∗ getHumPtr ()
  Get pointer to data raw_humidity.
- bool getTemp (uint16_t ∗temp)
  Get temperature data.
- bool getHumidity (uint16_t ∗hum)
  Get humidity data.
- void setValidity (bool validity)
  Set data val_validity.
- void updateLastValidValues ()
- uint8_t GetTempInteger ()
  Temperature formatting function - Integer part.
- uint8_t GetTempDecimal ()
  Temperature formatting function - Decimal part.
- uint8_t GetHumInteger ()
  Humidity formatting function - Integer part.
- uint8_t GetHumDecimal ()
  Humidity formatting function - Decimal part.
- bool GetValidity ()
  Data validity get function.
- bool updateTaskPeriod (uint16_t period)
  Task period update.
- uint16_t getTaskPeriod ()
  Task period get function.

Static Public Member Functions

- static void readTempSensor_task ()

  Task for reading temperature and humidity values.

Private Attributes

- uint16_t read_temperature
- uint16_t read_humidity
- bool validity_last_read
- bool validity
- uint32_t valid_pit
- uint16_t valid_temp
- uint16_t valid_hum
- uint16_t task_period

3.21.1 Detailed Description

Class for temperature sensor.

This class defines all functions used to read data from temperature sensor and monotor it

Definition at line 19 of file TempSensor.h.

3.21.2 Constructor & Destructor Documentation

3.21.2.1 TempSensor()

`TempSensor::TempSensor ( )`

Class constructor.

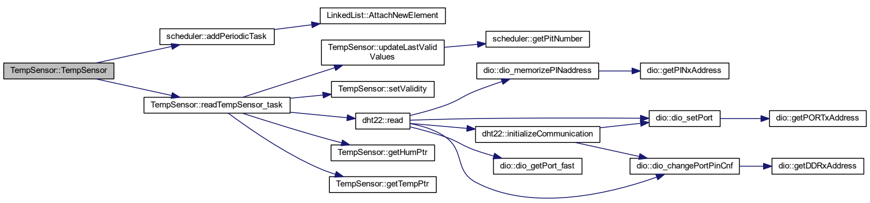
This function initializes all data of the class TempSensor

Returns

Nothing

Definition at line 37 of file TempSensor.cpp.

Here is the call graph for this function:



3.21.3 Member Function Documentation

3.21.3.1 GetHumDecimal()

```
uint8_t TempSensor::GetHumDecimal ( )  [inline]
```

Humidity formatting function - Decimal part.

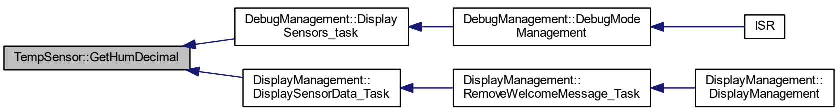
This function return the decimal part of the humidity

Returns

Decimal value of the humidity

Definition at line 122 of file TempSensor.h.

Here is the caller graph for this function:



3.21.3.2 getHumidity()

```
bool TempSensor::getHumidity (
            uint16_t * hum )
```

Get humidity data.

This function returns the value of the humidity. If the official value is not valid, the function return false.

Parameters

out	*hum*	Humidity value

Returns

Validity of humidity

Definition at line 89 of file TempSensor.cpp.

3.21.3.3 GetHumInteger()

```
uint8_t TempSensor::GetHumInteger ( ) [inline]
```

Humidity formatting function - Integer part.

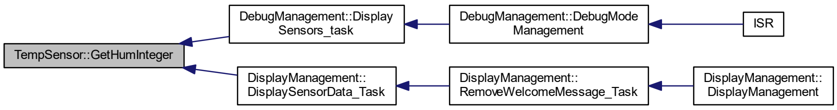
This function return the integer part of the humidity

Returns

Integer value of the humidity

Definition at line 111 of file TempSensor.h.

Here is the caller graph for this function:



3.21.3.4 getHumPtr()

```
uint16_t * TempSensor::getHumPtr ( )
```

Get pointer to data raw_humidity.

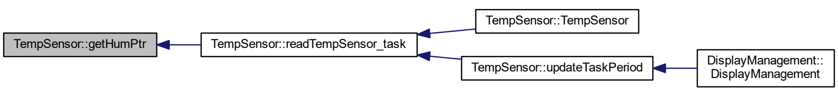
This function returns a pointer to the class member raw_humidity

Returns

Pointer to raw_humidity

Definition at line 64 of file TempSensor.cpp.

Here is the caller graph for this function:



3.21.3.5 getTaskPeriod()

```
uint16_t TempSensor::getTaskPeriod ( )  [inline]
```

Task period get function.

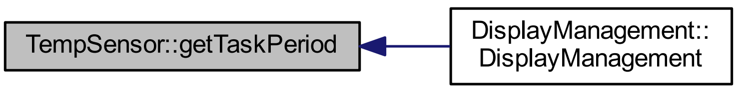
This function returns the period of the sensor task

Returns

Period of the task (ms)

Definition at line 153 of file TempSensor.h.

Here is the caller graph for this function:



3.21.3.6 getTemp()

```
bool TempSensor::getTemp (
            uint16_t * temp )
```

Get temperature data.

This function returns the value of the temperature. If the official value is not valid, the function return false.

Parameters

out	*temp*	Temperature value

Returns

Validity of temperature

Definition at line 95 of file TempSensor.cpp.

3.21.3.7 GetTempDecimal()

`uint8_t TempSensor::GetTempDecimal ( ) [inline]`

Temperature formatting function - Decimal part.

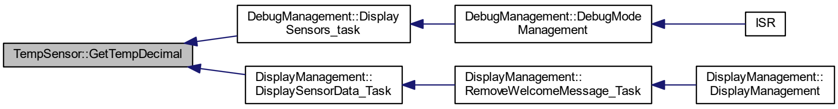
This function return the decimal part of the temperature

Returns

Decimal value of the temperature

Definition at line 100 of file TempSensor.h.

Here is the caller graph for this function:



3.21.3.8 GetTempInteger()

`uint8_t TempSensor::GetTempInteger ( ) [inline]`

Temperature formatting function - Integer part.

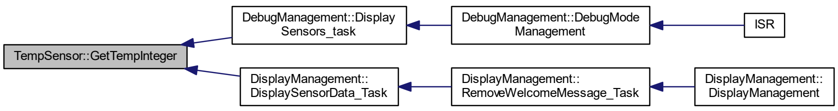
This function return the integer part of the temperature

Returns

Integer value of the temperature

Definition at line 89 of file TempSensor.h.

Here is the caller graph for this function:



3.21.3.9 getTempPtr()

```
uint16_t * TempSensor::getTempPtr ( )
```

Get pointer to data raw_temperature.

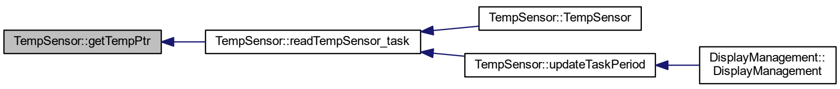
This function returns a pointer to the class member raw_temperature

Returns

Pointer to raw_temperature

Definition at line 69 of file TempSensor.cpp.

Here is the caller graph for this function:



3.21.3.10 GetValidity()

```
bool TempSensor::GetValidity ( ) [inline]
```

Data validity get function.

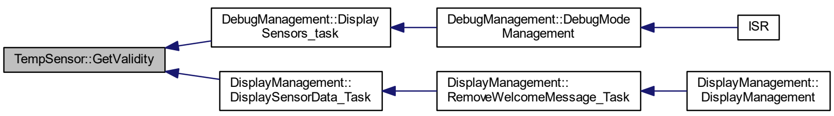
This function returns the validity of the sensor data

Returns

True if the sensor values are valid, false otherwise

Definition at line 133 of file TempSensor.h.

Here is the caller graph for this function:



3.21.3.11 readTempSensor_task()

```
void TempSensor::readTempSensor_task ( ) [static]
```

Task for reading temperature and humidity values.

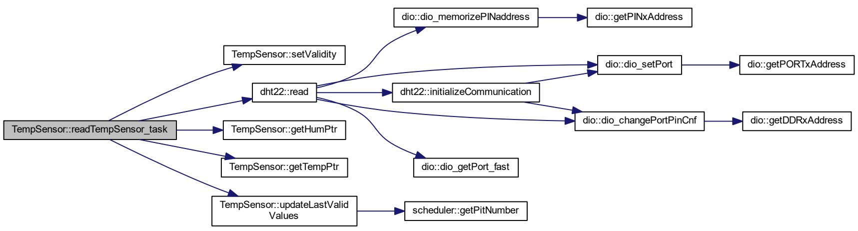
This task reads temperature and humidity data using DHT22 driver. It is called every 5 seconds.

Returns

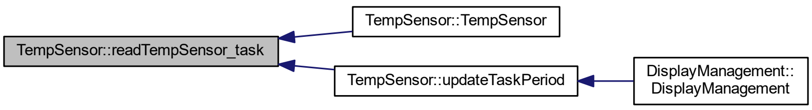
Nothing

Definition at line 53 of file TempSensor.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.21.3.12 setValidity()

```
void TempSensor::setValidity (
            bool validity )
```

Set data val_validity.

This function sets the class member val_validity

Parameters

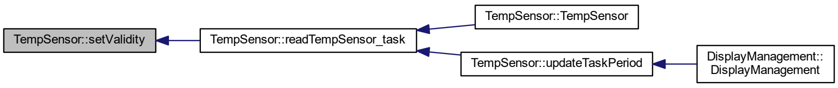
in	*validity*	Value of validity

Returns

Nothing

Definition at line 59 of file TempSensor.cpp.

Here is the caller graph for this function:

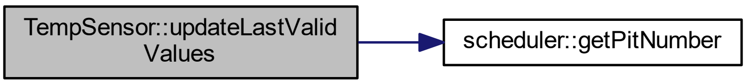


3.21.3.13 updateLastValidValues()

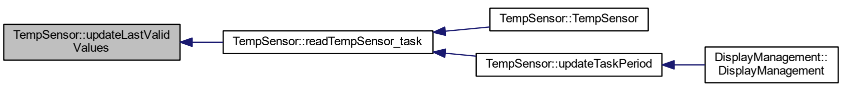
```
void TempSensor::updateLastValidValues ( )
```

Definition at line 74 of file TempSensor.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.21.3.14 updateTaskPeriod()

```
bool TempSensor::updateTaskPeriod (
            uint16_t period )
```

Task period update.

This function updates the period of the temperature task.

Parameters

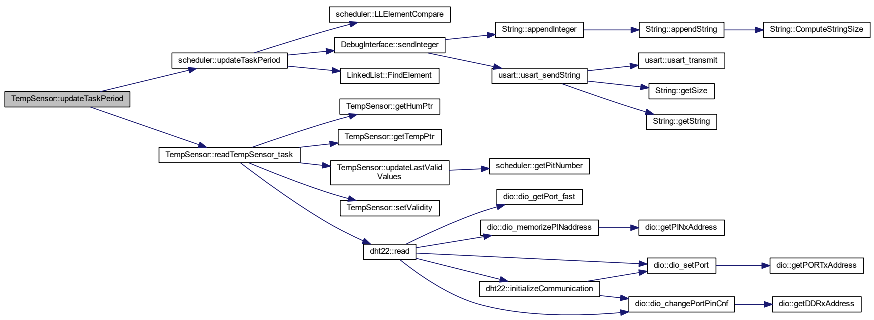
in	*period*	New period of the task

Returns

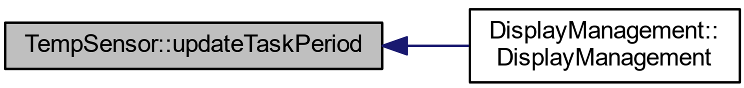
True if the period has been updated, false otherwise

Definition at line 101 of file TempSensor.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.21.4 Member Data Documentation

3.21.4.1 read_humidity

`uint16_t TempSensor::read_humidity [private]`

Raw value of humidity read from DHT22 (= real humidity ∗10)

Definition at line 161 of file TempSensor.h.

3.21.4.2 read_temperature

```
uint16_t TempSensor::read_temperature  [private]
```

Raw value of temperature read from DHT22 (= real temperature $*10$)

Definition at line 160 of file TempSensor.h.

3.21.4.3 task_period

```
uint16_t TempSensor::task_period  [private]
```

Task period

Definition at line 170 of file TempSensor.h.

3.21.4.4 valid_hum

```
uint16_t TempSensor::valid_hum  [private]
```

Valid value of humidity

Definition at line 168 of file TempSensor.h.

3.21.4.5 valid_pit

```
uint32_t TempSensor::valid_pit  [private]
```

pit number of the last time when data were valid

Definition at line 165 of file TempSensor.h.

3.21.4.6 valid_temp

```
uint16_t TempSensor::valid_temp  [private]
```

Valid value of temperature

Definition at line 167 of file TempSensor.h.

3.21.4.7 validity

```
bool TempSensor::validity [private]
```

validity of official temperature and humidity data

Definition at line 164 of file TempSensor.h.

3.21.4.8 validity_last_read

```
bool TempSensor::validity_last_read [private]
```

Validity of last read temperature and humidity data

Definition at line 162 of file TempSensor.h.

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

- TempSensor.h
- TempSensor.cpp

3.22 timer Class Reference

Class defining a timer.

```
#include <timer.h>
```

Public Member Functions

- timer ()
  Class constructor.
- void configureTimer1 (uint16_t a_prescaler, uint16_t a_ctcValue)
  Configures Timer #1.
- void startTimer1 ()
  Start Timer #1.
- void stopTimer1 ()
  Stops Timer #1.
- uint16_t getTimer1Value ()
  Reads current value of timer #1.

Private Attributes

- uint8_t prescaler

3.22.1 Detailed Description

Class defining a timer.

This class defines a timer/counter. The selected timer is configured in CTC mode and interrupts are enabled. The prescaler value and CTC value can both be configured by user.

Definition at line 19 of file timer.h.

3.22.2 Constructor & Destructor Documentation

3.22.2.1 timer()

```
timer::timer ( )
```

Class constructor.

This function initializes class attributes

Returns

Nothing

Definition at line 13 of file timer.cpp.

3.22.3 Member Function Documentation

3.22.3.1 configureTimer1()

```
void timer::configureTimer1 (
            uint16_t a_prescaler,
            uint16_t a_ctcValue )
```

Configures Timer #1.

This function configures hardware timer #1 in CTC mode, enables its interrupts, sets prescaler to a_prescaler and CTC value to a_ctcValue

Parameters

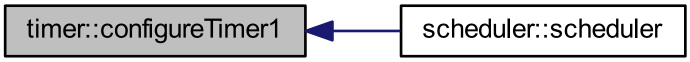
in	*a_prescaler*	prescaler value
in	*a_ctcValue*	Value to which the counter will compare before raising an interrupt

Returns

Nothing

Definition at line 18 of file timer.cpp.

Here is the caller graph for this function:



3.22.3.2 getTimer1Value()

```
uint16_t timer::getTimer1Value ( ) [inline]
```

Reads current value of timer #1.

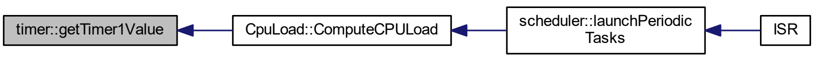
This function reads the value of of timer #1 using register TCNT1. The function is inlined to speed up SW execution.

Returns

Current timer value

Definition at line 58 of file timer.h.

Here is the caller graph for this function:



3.22.3.3 startTimer1()

```
void timer::startTimer1 ( )
```

Start Timer #1.

This functions starts Timer #1. Timer shall be initialized before this function is called.

Returns

Nothing

Definition at line 56 of file timer.cpp.

Here is the caller graph for this function:

3.22.3.4 stopTimer1()

```
void timer::stopTimer1 ( )
```

Stops Timer #1.

This functions stops timer #1 by resetting bits 0-2 of TCCR1B

Returns

Nothing

Definition at line 67 of file timer.cpp.

3.22.4 Member Data Documentation

3.22.4.1 prescaler

```
uint8_t timer::prescaler [private]
```

Definition at line 64 of file timer.h.

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

- timer.h
- timer.cpp

3.23 usart Class Reference

USART serial bus class.

```
#include <usart.h>
```

Public Member Functions

- usart (uint16_t a_BaudRate)

  Class usart constructor.
- void usart_sendString (String ∗str)

  Send a string on USART link.
- void usart_sendByte (uint8_t data)

  Send a single byte on USART link.
- void setBaudRate (uint16_t a_BaudRate)

  Setting baud rate.
- void usart_init ()

  USART hardware initialization.
- uint8_t usart_read ()

  USART read function.

Private Member Functions

- void usart_transmit (uint8_t Data)

  USART Transmit data.

Private Attributes

- uint16_t BaudRate

3.23.1 Detailed Description

USART serial bus class.

This class defines all useful functions for USART serial bus

Definition at line 16 of file usart.h.

3.23.2 Constructor & Destructor Documentation

3.23.2.1 usart()

```
usart::usart (
            uint16_t a_BaudRate )
```

Class usart constructor.

Initializes the class and call hardware initialization function

Parameters

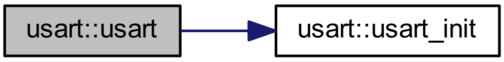
in	*a_BaudRate*	Desired Baud Rate (16 bit) - up to 57600

Returns

Nothing.

Definition at line 17 of file usart.cpp.

Here is the call graph for this function:



3.23.3 Member Function Documentation

3.23.3.1 setBaudRate()

```
void usart::setBaudRate (
            uint16_t a_BaudRate )  [inline]
```

Setting baud rate.

This function sets the attribute BaudRate of the class usart

Parameters

in	*a_BaudRate*	Desired Baud Rate (16 bit) - up to 57600

Returns

Nothing

Definition at line 73 of file usart.cpp.

3.23.3.2 usart_init()

```
void usart::usart_init ( )
```

USART hardware initialization.

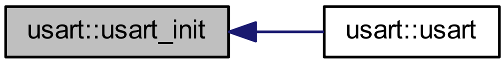
This function will initialize the USART using selected baudrate. User must pay attention to select one of the usually used Baud Rate (9600, 19200, 38400, 57600). Note that since an uint16 is used as argument, Baud rate cannot be more than 57600.

Returns

Nothing.

Definition at line 24 of file usart.cpp.

Here is the caller graph for this function:



3.23.3.3 usart_read()

```
uint8_t usart::usart_read ( )
```

USART read function.

This function will read reception register of USART

Returns

The function returns the 8 bits read from reception buffer

Definition at line 89 of file usart.cpp.

3.23.3.4 usart_sendByte()

```
void usart::usart_sendByte (
            uint8_t data )
```

Send a single byte on USART link.

This function writes the given byte to the serial link using usart_transmit function

Parameters

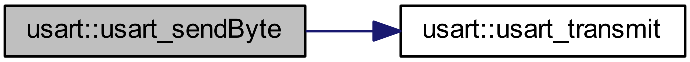
in	*data*	Data byte being sent

Returns

Nothing.

Definition at line 67 of file usart.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

3.23.3.5 usart_sendString()

```
void usart::usart_sendString (
            String * str )
```

Send a string on USART link.

This function writes the string object data to the serial link using usart_transmit function

Parameters

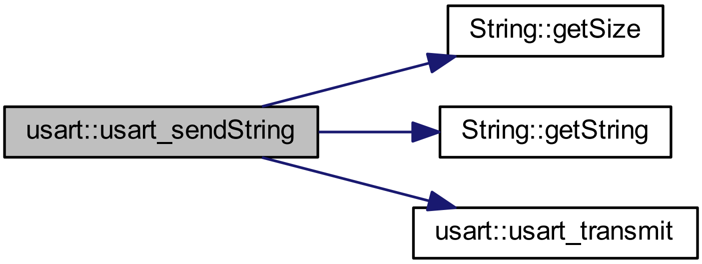
in	*str*	Pointer to the string being sent

Returns

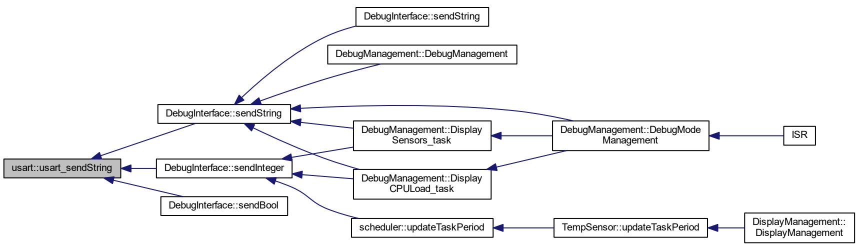
Nothing.

Definition at line 47 of file usart.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.23.3.6 usart_transmit()

```
void usart::usart_transmit (
            uint8_t Data )  [private]
```

USART Transmit data.

Nothing Special. It just wait for the transmit buffer is empty before writing it again.

Parameters

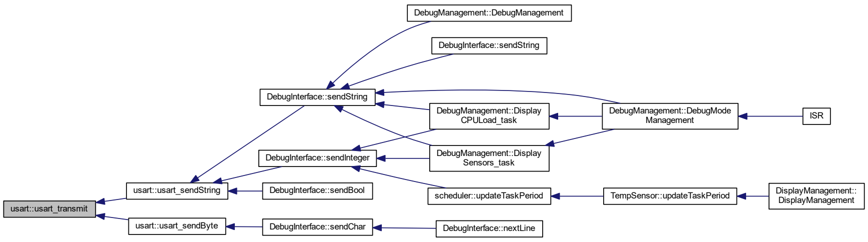
in	*Data*	Desired data char to transmit

Returns

Nothing.

Definition at line 80 of file usart.cpp.

Here is the caller graph for this function:



3.23.4 Member Data Documentation

3.23.4.1 BaudRate

`uint16_t usart::BaudRate [private]`

Defines the baud rate used by driver

Definition at line 77 of file usart.h.

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

- usart.h
- usart.cpp

Chapter 4

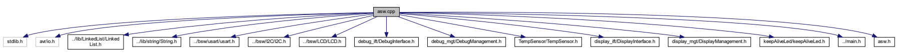
File Documentation

4.1 asw.cpp File Reference

ASW main file.

```
#include <stdlib.h>
#include <avr/io.h>
#include "../lib/LinkedList/LinkedList.h"
#include "../lib/string/String.h"
#include "../bsw/usart/usart.h"
#include "../bsw/I2C/I2C.h"
#include "../bsw/LCD/LCD.h"
#include "debug_ift/DebugInterface.h"
#include "debug_mgt/DebugManagement.h"
#include "TempSensor/TempSensor.h"
#include "display_ift/DisplayInterface.h"
#include "display_mgt/DisplayManagement.h"
#include "keepAliveLed/keepAliveLed.h"
#include "../main.h"
#include "asw.h"
```

Include dependency graph for asw.cpp:



Functions

- void asw_init ()

    Initialization of ASW.

Variables

- T_ASW_cnf_struct ASW_cnf_struct

4.1.1 Detailed Description

ASW main file.

Date

15 mars 2018

Author

nicls67

4.1.2 Function Documentation

4.1.2.1 asw_init()

```
void asw_init ( )
```

Initialization of ASW.

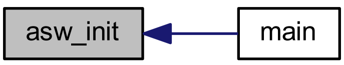
This function instantiates all applicative objects. The addresses of objects are then stored in ASW_cnf_struct structure. This function shall be called after BSW initialization function.

Returns

Nothing

Definition at line 37 of file asw.cpp.

Here is the caller graph for this function:



4.1.3 Variable Documentation

4.1.3.1 ASW_cnf_struct

`T_ASW_cnf_struct ASW_cnf_struct`

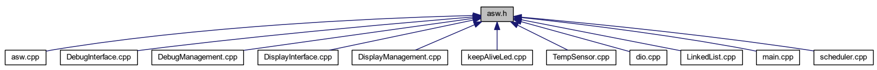
ASW configuration structure

Definition at line 34 of file asw.cpp.

4.2 asw.h File Reference

ASW main header file.

This graph shows which files directly or indirectly include this file:



Classes

- struct T_ASW_cnf_struct

    ASW configuration structure.

Functions

- void asw_init ()

    Initialization of ASW.

Variables

- T_ASW_cnf_struct ASW_cnf_struct

4.2.1 Detailed Description

ASW main header file.

Date

15 mars 2018

Author

nicls67

4.2.2 Function Documentation

4.2.2.1 asw_init()

```
void asw_init ( )
```

Initialization of ASW.

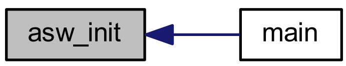
This function instantiates all applicative objects. The addresses of objects are then stored in ASW_cnf_struct structure. This function shall be called after BSW initialization function.

Returns

Nothing

Definition at line 37 of file asw.cpp.

Here is the caller graph for this function:



4.2.3 Variable Documentation

4.2.3.1 ASW_cnf_struct

```
T_ASW_cnf_struct ASW_cnf_struct
```

ASW configuration structure

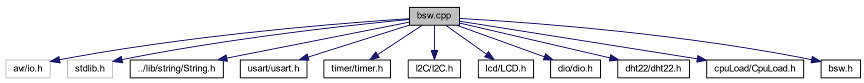
Definition at line 34 of file asw.cpp.

4.3 bsw.cpp File Reference

BSW main file.

```
#include <avr/io.h>
#include <stdlib.h>
#include "../lib/string/String.h"
#include "usart/usart.h"
#include "timer/timer.h"
#include "I2C/I2C.h"
#include "lcd/LCD.h"
#include "dio/dio.h"
#include "dht22/dht22.h"
#include "cpuLoad/CpuLoad.h"
#include "bsw.h"
```

Include dependency graph for bsw.cpp:



Functions

- void bsw_init ()

  Initialization of BSW.

Variables

- T_BSW_cnf_struct BSW_cnf_struct

4.3.1 Detailed Description

BSW main file.

Date

13 mars 2018

Author

nicls67

4.3.2 Function Documentation

4.3.2.1 bsw_init()

```
void bsw_init ( )
```

Initialization of BSW.

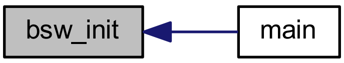
This function instantiates all driver objects, leading hardware initialization. The addresses of driver objects are then stored in BSW_cnf_struct structure.

Returns

Nothing

Definition at line 26 of file bsw.cpp.

Here is the caller graph for this function:



4.3.3 Variable Documentation

4.3.3.1 BSW_cnf_struct

```
T_BSW_cnf_struct BSW_cnf_struct
```

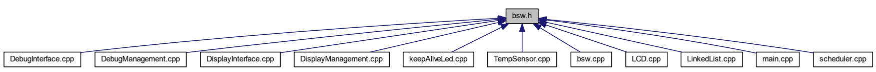
BSW configuration structure

Definition at line 24 of file bsw.cpp.

4.4 bsw.h File Reference

BSW main header file.

This graph shows which files directly or indirectly include this file:



Classes

- struct T_BSW_cnf_struct
  BSW configuration structure.

Functions

- void bsw_init ()
  Initialization of BSW.

Variables

- T_BSW_cnf_struct BSW_cnf_struct

4.4.1 Detailed Description

BSW main header file.

Date

13 mars 2018

Author

nicls67

4.4.2 Function Documentation

4.4.2.1 bsw_init()

```
void bsw_init ( )
```

Initialization of BSW.

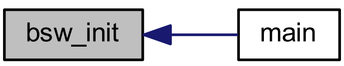
This function instantiates all driver objects, leading hardware initialization. The addresses of driver objects are then stored in BSW_cnf_struct structure.

Returns

Nothing

Definition at line 26 of file bsw.cpp.

Here is the caller graph for this function:



4.4.3 Variable Documentation

4.4.3.1 BSW_cnf_struct

`T_BSW_cnf_struct BSW_cnf_struct`

BSW configuration structure

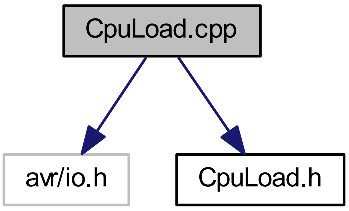
Definition at line 24 of file bsw.cpp.

4.5 CpuLoad.cpp File Reference

Defines functions of class CpuLoad.

```
#include <avr/io.h>
#include "CpuLoad.h"
```

Include dependency graph for CpuLoad.cpp:



4.5.1 Detailed Description

Defines functions of class CpuLoad.

Date

21 mars 2019

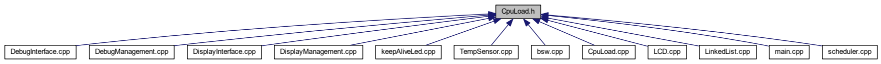
Author

nicls67

4.6 CpuLoad.h File Reference

CpuLoad class header file.

This graph shows which files directly or indirectly include this file:



Classes

- class CpuLoad

  Class defining CPU load libraries.

Macros

- #define NB_OF_SAMPLES 50

4.6.1 Detailed Description

CpuLoad class header file.

Date

21 mars 2019

Author

nicls67

4.6.2 Macro Definition Documentation

4.6.2.1 NB_OF_SAMPLES

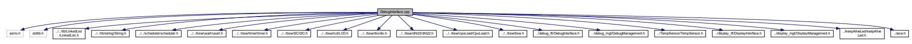
```
#define NB_OF_SAMPLES 50
```

Definition at line 13 of file CpuLoad.h.

4.7 DebugInterface.cpp File Reference

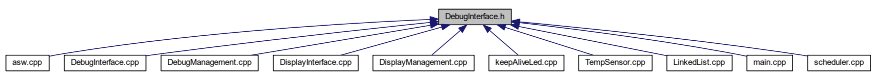
```
#include <avr/io.h>
#include <stdlib.h>
#include "../../lib/LinkedList/LinkedList.h"
#include "../../lib/string/String.h"
#include "../../scheduler/scheduler.h"
#include "../../bsw/usart/usart.h"
#include "../../bsw/timer/timer.h"
#include "../../bsw/I2C/I2C.h"
#include "../../bsw/lcd/LCD.h"
#include "../../bsw/dio/dio.h"
#include "../../bsw/dht22/dht22.h"
#include "../../bsw/cpuLoad/CpuLoad.h"
#include "../../bsw/bsw.h"
#include "../debug_ift/DebugInterface.h"
#include "../debug_mgt/DebugManagement.h"
#include "../TempSensor/TempSensor.h"
#include "../display_ift/DisplayInterface.h"
#include "../display_mgt/DisplayManagement.h"
#include "../keepAliveLed/keepAliveLed.h"
#include "../asw.h"
```

Include dependency graph for DebugInterface.cpp:



4.8 DebugInterface.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class DebugInterface

  Class used for debugging on usart link.

Macros

- #define USART_BAUDRATE (uint16_t)9600

4.8.1 Macro Definition Documentation

4.8.1.1 USART_BAUDRATE

```
#define USART_BAUDRATE (uint16_t)9600
```

usart connection to PC uses a baud rate of 9600

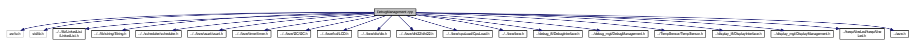
Definition at line 15 of file DebugInterface.h.

4.9 DebugManagement.cpp File Reference

Debug management class source file.

```
#include <avr/io.h>
#include <stdlib.h>
#include "../../lib/LinkedList/LinkedList.h"
#include "../../lib/string/String.h"
#include "../../scheduler/scheduler.h"
#include "../../bsw/usart/usart.h"
#include "../../bsw/timer/timer.h"
#include "../../bsw/I2C/I2C.h"
#include "../../bsw/lcd/LCD.h"
#include "../../bsw/dio/dio.h"
#include "../../bsw/dht22/dht22.h"
#include "../../bsw/cpuLoad/CpuLoad.h"
#include "../../bsw/bsw.h"
#include "../debug_ift/DebugInterface.h"
#include "../debug_mgt/DebugManagement.h"
#include "../TempSensor/TempSensor.h"
#include "../display_ift/DisplayInterface.h"
#include "../display_mgt/DisplayManagement.h"
#include "../keepAliveLed/keepAliveLed.h"
#include "../asw.h"
```

Include dependency graph for DebugManagement.cpp:



Variables

- const uint8_t str_debug_main_menu [ ]

  Main menu of debug mode.

4.9.1 Detailed Description

Debug management class source file.

Date

8 mai 2019

Author

nicls67

4.9.2 Variable Documentation

4.9.2.1 str_debug_main_menu

```
const uint8_t str_debug_main_menu[]
```

Initial value:

```
=
        "\n\n"
        "Menu principal :  \n"
        "1 : Afficher donnees capteurs\n"
        "2 : Afficher charge CPU\n"
        "\n"
        "s : Quitter debug\n"
```

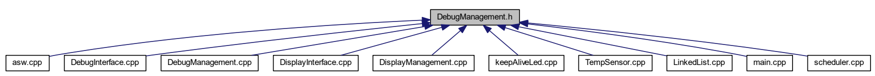
Main menu of debug mode.

Definition at line 41 of file DebugManagement.cpp.

4.10 DebugManagement.h File Reference

Debug management class header file.

This graph shows which files directly or indirectly include this file:



Classes

- class DebugManagement

    Debug management class.

Macros

- #define PERIOD_MS_TASK_DISPLAY_SENSORS 5000
- #define PERIOD_MS_TASK_DISPLAY_CPU_LOAD 5000

Enumerations

- enum debug_state_t { INIT, DISPLAY_DATA, DISPLAY_CPU_LOAD }

    Defines the debug states.

4.10.1 Detailed Description

Debug management class header file.

Date

8 mai 2019

Author

nicls67

4.10.2 Macro Definition Documentation

4.10.2.1 PERIOD_MS_TASK_DISPLAY_CPU_LOAD

```
#define PERIOD_MS_TASK_DISPLAY_CPU_LOAD 5000
```

Period for displaying CPU load data

Definition at line 14 of file DebugManagement.h.

4.10.2.2 PERIOD_MS_TASK_DISPLAY_SENSORS

```
#define PERIOD_MS_TASK_DISPLAY_SENSORS 5000
```

Period for displaying temperature and humidity data

Definition at line 13 of file DebugManagement.h.

4.10.3 Enumeration Type Documentation

4.10.3.1 debug_state_t

```
enum debug_state_t
```

Defines the debug states.

Enumerator

INIT	Init state : main menu has been displayed, wait for a received character
DISPLAY_DATA	Display sensor data in continuous
DISPLAY_CPU_LOAD	Display CPU load in continuous

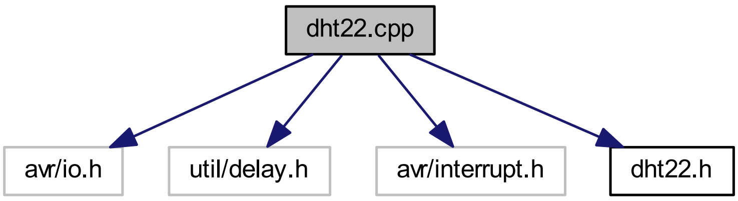
Definition at line 19 of file DebugManagement.h.

4.11 dht22.cpp File Reference

This file defines classes for DHT22 driver.

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "dht22.h"
```

Include dependency graph for dht22.cpp:



Macros

- #define MAX_WAIT_TIME_US 100

4.11.1 Detailed Description

This file defines classes for DHT22 driver.

Date

23 mars 2018

Author

nicls67

4.11.2 Macro Definition Documentation

4.11.2.1 MAX_WAIT_TIME_US

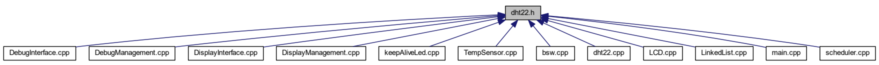
```
#define MAX_WAIT_TIME_US 100
```

Definition at line 20 of file dht22.cpp.

4.12 dht22.h File Reference

DHT22 driver header file.

This graph shows which files directly or indirectly include this file:



Classes

- class dht22

  DHT 22 driver class.

Macros

- #define DHT22_PORT ENCODE_PORT(PORT_B, 6)

4.12.1 Detailed Description

DHT22 driver header file.

Date

23 mars 2018

Author

nicls67

4.12.2 Macro Definition Documentation

4.12.2.1 DHT22_PORT

```
#define DHT22_PORT ENCODE_PORT(PORT_B, 6)
```

DHT22 is connected to port PB6

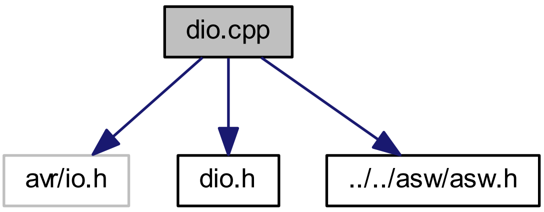
Definition at line 16 of file dht22.h.

4.13 dio.cpp File Reference

DIO library.

```
#include <avr/io.h>
#include "dio.h"
#include "../../asw/asw.h"
```

Include dependency graph for dio.cpp:



4.13.1 Detailed Description

DIO library.

Date

13 mars 2018

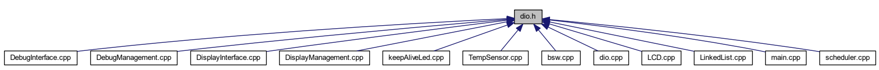
Author

nicls67

4.14 dio.h File Reference

DIO library header file.

This graph shows which files directly or indirectly include this file:



Classes

- class dio

  DIO class.

Macros

- #define PORT_CNF_OUT 1
- #define PORT_CNF_IN 0
- #define ENCODE_PORT(port, pin) (uint8_t)((((uint8_t)(port & 0xF)) << 3) | (uint8_t)(pin & 0x7))
- #define DECODE_PORT(portcode) (uint8_t)((portcode >> 3) & 0xF)
- #define DECODE_PIN(portcode) (uint8_t)(portcode & 0x7)
- #define PORT_A 0
- #define PORT_B 1
- #define PORT_C 2
- #define PORT_D 3

4.14.1 Detailed Description

DIO library header file.

Date

13 mars 2018

Author

nicls67

4.14.2 Macro Definition Documentation

4.14.2.1 DECODE_PIN

```
#define DECODE_PIN(
              portcode ) (uint8_t)(portcode & 0x7)
```

Macro used to extract pin index

Definition at line 19 of file dio.h.

4.14.2.2 DECODE_PORT

```
#define DECODE_PORT(
              portcode ) (uint8_t)((portcode >> 3) & 0xF)
```

Macro used to extract port index

Definition at line 18 of file dio.h.

4.14.2.3 ENCODE_PORT

```
#define ENCODE_PORT(
              port,
              pin ) (uint8_t)((((uint8_t)(port & 0xF)) << 3) | (uint8_t)(pin & 0x7))
```

Macro used to encode port and pin indexes into one single byte

Definition at line 17 of file dio.h.

4.14.2.4 PORT_A

```
#define PORT_A 0
```

PORTA index

Definition at line 21 of file dio.h.

4.14.2.5 PORT_B

```
#define PORT_B 1
```

PORTB index

Definition at line 22 of file dio.h.

4.14.2.6 PORT_C

```
#define PORT_C 2
```

PORTC index

Definition at line 23 of file dio.h.

4.14.2.7 PORT_CNF_IN

```
#define PORT_CNF_IN 0
```

Pin is configured as input

Definition at line 15 of file dio.h.

4.14.2.8 PORT_CNF_OUT

```
#define PORT_CNF_OUT 1
```

Pin is configured as output

Definition at line 14 of file dio.h.

4.14.2.9 PORT_D

```
#define PORT_D 3
```

PORTD index

Definition at line 24 of file dio.h.

4.15 dio_port_cnf.h File Reference

Digital ports configuration file.

Macros

- #define PORTB_CNF_DDRB (uint8_t)0b11000000

    Defines the configuration of DDRB register.
- #define PORTB_CNF_PORTB (uint8_t)0b11000000

    Defines the configuration of PORTB register.

4.15.1 Detailed Description

Digital ports configuration file.

Date

19 mars 2019

Author

nicls67

4.15.2 Macro Definition Documentation

4.15.2.1 PORTB_CNF_DDRB

```
#define PORTB_CNF_DDRB (uint8_t)0b11000000
```

Defines the configuration of DDRB register.

This constant defines the direction of IO pins of PORT B. It will configure register DDRB.
PB0 : N/A
PB1 : N/A
PB2 : N/A
PB3 : N/A
PB4 : N/A
PB5 : N/A
PB6 : OUT
PB7 : OUT

Definition at line 25 of file dio_port_cnf.h.

4.15.2.2 PORTB_CNF_PORTB

```
#define PORTB_CNF_PORTB (uint8_t)0b11000000
```

Defines the configuration of PORTB register.

This constant defines the initial value of IO pins for PORT B. It will configure register PORTB. Pins configured as input shall not be configured here.
PB0 : N/A
PB1 : N/A
PB2 : N/A
PB3 : N/A
PB4 : N/A
PB5 : N/A
PB6 : HIGH
PB7 : HIGH

Definition at line 40 of file dio_port_cnf.h.

4.16 dio_reg_atm2560.h File Reference

Macros

- #define PORTA_PTR (volatile uint8_t ∗)(0x02 + 0x20)
- #define PORTB_PTR (volatile uint8_t ∗)(0x05 + 0x20)
- #define PORTC_PTR (volatile uint8_t ∗)(0x08 + 0x20)
- #define PORTD_PTR (volatile uint8_t ∗)(0x0B + 0x20)
- #define PINA_PTR (volatile uint8_t ∗)(0x00 + 0x20)
- #define PINB_PTR (volatile uint8_t ∗)(0x03 + 0x20)
- #define PINC_PTR (volatile uint8_t ∗)(0x06 + 0x20)
- #define PIND_PTR (volatile uint8_t ∗)(0x09 + 0x20)
- #define DDRA_PTR (volatile uint8_t ∗)(0x01 + 0x20)
- #define DDRB_PTR (volatile uint8_t ∗)(0x04 + 0x20)
- #define DDRC_PTR (volatile uint8_t ∗)(0x07 + 0x20)
- #define DDRD_PTR (volatile uint8_t ∗)(0x0A + 0x20)

4.16.1 Macro Definition Documentation

4.16.1.1 DDRA_PTR

```
#define DDRA_PTR (volatile uint8_t *)(0x01 + 0x20)
```

Macro defining pointer to DDR A register

Definition at line 24 of file dio_reg_atm2560.h.

4.16.1.2 DDRB_PTR

```
#define DDRB_PTR (volatile uint8_t *)(0x04 + 0x20)
```

Macro defining pointer to DDR B register

Definition at line 25 of file dio_reg_atm2560.h.

4.16.1.3 DDRC_PTR

```
#define DDRC_PTR (volatile uint8_t *)(0x07 + 0x20)
```

Macro defining pointer to DDR C register

Definition at line 26 of file dio_reg_atm2560.h.

4.16.1.4 DDRD_PTR

```
#define DDRD_PTR (volatile uint8_t *)(0x0A + 0x20)
```

Macro defining pointer to DDR D register

Definition at line 27 of file dio_reg_atm2560.h.

4.16.1.5 PINA_PTR

```
#define PINA_PTR (volatile uint8_t *)(0x00 + 0x20)
```

Macro defining pointer to PIN A register

Definition at line 19 of file dio_reg_atm2560.h.

4.16.1.6 PINB_PTR

```
#define PINB_PTR (volatile uint8_t *)(0x03 + 0x20)
```

Macro defining pointer to PIN B register

Definition at line 20 of file dio_reg_atm2560.h.

4.16.1.7 PINC_PTR

```
#define PINC_PTR (volatile uint8_t *)(0x06 + 0x20)
```

Macro defining pointer to PIN C register

Definition at line 21 of file dio_reg_atm2560.h.

4.16.1.8 PIND_PTR

```
#define PIND_PTR (volatile uint8_t *)(0x09 + 0x20)
```

Macro defining pointer to PIN D register

Definition at line 22 of file dio_reg_atm2560.h.

4.16.1.9 PORTA_PTR

```
#define PORTA_PTR (volatile uint8_t *)(0x02 + 0x20)
```

Macro defining pointer to PORT A register

Definition at line 14 of file dio_reg_atm2560.h.

4.16.1.10 PORTB_PTR

```
#define PORTB_PTR (volatile uint8_t *)(0x05 + 0x20)
```

Macro defining pointer to PORT B register

Definition at line 15 of file dio_reg_atm2560.h.

4.16.1.11 PORTC_PTR

```
#define PORTC_PTR (volatile uint8_t *)(0x08 + 0x20)
```

Macro defining pointer to PORT C register

Definition at line 16 of file dio_reg_atm2560.h.

4.16.1.12 PORTD_PTR

```
#define PORTD_PTR (volatile uint8_t *)(0x0B + 0x20)
```

Macro defining pointer to PORT D register

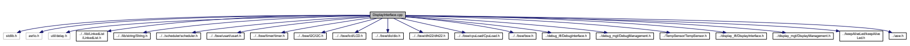
Definition at line 17 of file dio_reg_atm2560.h.

4.17 DisplayInterface.cpp File Reference

Source code file for display services.

```
#include <stdlib.h>
#include <avr/io.h>
#include <util/delay.h>
#include "../../lib/LinkedList/LinkedList.h"
#include "../../lib/string/String.h"
#include "../../scheduler/scheduler.h"
#include "../../bsw/usart/usart.h"
#include "../../bsw/timer/timer.h"
#include "../../bsw/I2C/I2C.h"
#include "../../bsw/lcd/LCD.h"
#include "../../bsw/dio/dio.h"
#include "../../bsw/dht22/dht22.h"
#include "../../bsw/cpuLoad/CpuLoad.h"
#include "../../bsw/bsw.h"
#include "../debug_ift/DebugInterface.h"
#include "../debug_mgt/DebugManagement.h"
#include "../TempSensor/TempSensor.h"
#include "../display_ift/DisplayInterface.h"
#include "../display_mgt/DisplayManagement.h"
#include "../keepAliveLed/keepAliveLed.h"
#include "../asw.h"
```

Include dependency graph for DisplayInterface.cpp:



4.17.1 Detailed Description

Source code file for display services.

Date

23 avr. 2019

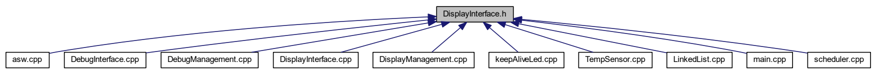
Author

nicls67

4.18 DisplayInterface.h File Reference

DisplayInterface class header file.

This graph shows which files directly or indirectly include this file:



Classes

- struct T_Display_shift_data

  Structure containing shift data.
- struct T_display_data

  Structure containing display data.
- class DisplayInterface

  Display interface services class.

Macros

- #define DISPLAY_LINE_SHIFT_PERIOD_MS 500
- #define DISPLAY_LINE_SHIFT_TEMPO_TIME 6

Enumerations

- enum T_DisplayInterface_LineDisplayMode { NORMAL, LINE_SHIFT, GO_TO_NEXT_LINE }

  Modes for line display.
- enum T_DisplayInterface_LineAlignment { LEFT, CENTER, RIGHT }

  Alignment mode for line display.

4.18.1 Detailed Description

DisplayInterface class header file.

Date

23 avr. 2019

Author

nicls67

4.18.2 Macro Definition Documentation

4.18.2.1 DISPLAY_LINE_SHIFT_PERIOD_MS

```
#define DISPLAY_LINE_SHIFT_PERIOD_MS 500
```

In "line shift" mode for line display, line is shifted every 500 ms

Definition at line 68 of file DisplayInterface.h.

4.18.2.2 DISPLAY_LINE_SHIFT_TEMPO_TIME

```
#define DISPLAY_LINE_SHIFT_TEMPO_TIME 6
```

In "line shift" mode for line display, a temporization of 6 periods is added at the end and the beginning of the lines

Definition at line 69 of file DisplayInterface.h.

4.18.3 Enumeration Type Documentation

4.18.3.1 T_DisplayInterface_LineAlignment

```
enum T_DisplayInterface_LineAlignment
```

Alignment mode for line display.

This enumeration defines the possible alignment mode for the text displayed. It is only used when the display mode is NORMAL or GO_TO_NEXT_LINE.

Enumerator

LEFT	Text is aligned left
CENTER	Text is centered
RIGHT	Text is aligned right

Definition at line 33 of file DisplayInterface.h.

4.18.3.2 T_DisplayInterface_LineDisplayMode

```
enum T_DisplayInterface_LineDisplayMode
```

Modes for line display.

This enumeration defines the available modes for the line display functionality :
1- Normal mode : if the string is too long to be displayed entirely, the end of the string is cut.
2- Line shift mode : the display is moving to display all the string.
3- Next line mode : the remaining characters are displayed on the next lines.

Enumerator

NORMAL	
LINE_SHIFT	
GO_TO_NEXT_LINE	

Definition at line 20 of file DisplayInterface.h.

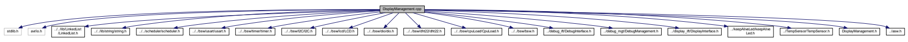
4.19 DisplayManagement.cpp File Reference

Display management source file.

```
#include <stdlib.h>
#include <avr/io.h>
#include "../../lib/LinkedList/LinkedList.h"
#include "../../lib/string/string.h"
#include "../../scheduler/scheduler.h"
#include "../../bsw/usart/usart.h"
#include "../../bsw/timer/timer.h"
#include "../../bsw/I2C/I2C.h"
#include "../../bsw/lcd/LCD.h"
#include "../../bsw/dio/dio.h"
#include "../../bsw/dht22/dht22.h"
#include "../../bsw/cpuLoad/CpuLoad.h"
#include "../../bsw/bsw.h"
#include "../debug_ift/DebugInterface.h"
#include "../debug_mgt/DebugManagement.h"
#include "../display_ift/DisplayInterface.h"
```

```
#include "../keepAliveLed/keepAliveLed.h"
#include "../TempSensor/TempSensor.h"
#include "DisplayManagement.h"
#include "../asw.h"
```

Include dependency graph for DisplayManagement.cpp:



4.19.1 Detailed Description

Display management source file.

Date

1 mai 2019

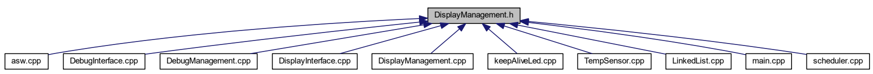
Author

nicls67

4.20 DisplayManagement.h File Reference

Display management class header file.

This graph shows which files directly or indirectly include this file:



Classes

- class DisplayManagement

    Display management class.

Macros

- #define DISPLAY_MGT_LCD_I2C_ADDR 0x27
- #define DISPLAY_MGT_PERIOD_TASK_SENSOR 1500
- #define DISPLAY_MGT_PERIOD_WELCOME_MSG_REMOVAL 5000
- #define DISPLAY_MGT_LINE_TEMP 0
- #define DISPLAY_MGT_LINE_HUM 1
- #define DISPLAY_MGT_I2C_BITRATE (uint32_t)100000

Variables

- const T_LCD_conf_struct LCD_init_cnf

    LCD configuration structure.
- const uint8_t welcomeMessageString [ ] = "Bienvenue !"
- const uint8_t tempDisplayString [ ] = "Temperature : "
- const uint8_t humidityDisplayString [ ] = "Humidite : "

4.20.1 Detailed Description

Display management class header file.

Date

1 mai 2019

Author

nicls67

4.20.2 Macro Definition Documentation

4.20.2.1 DISPLAY_MGT_I2C_BITRATE

```
#define DISPLAY_MGT_I2C_BITRATE (uint32_t)100000
```

I2C bus bitrate is 100 kHz

Definition at line 21 of file DisplayManagement.h.

4.20.2.2 DISPLAY_MGT_LCD_I2C_ADDR

```
#define DISPLAY_MGT_LCD_I2C_ADDR 0x27
```

I2C address of the screen

Definition at line 13 of file DisplayManagement.h.

4.20.2.3 DISPLAY_MGT_LINE_HUM

```
#define DISPLAY_MGT_LINE_HUM 1
```

Current humidity is displayed on line 1

Definition at line 19 of file DisplayManagement.h.

4.20.2.4 DISPLAY_MGT_LINE_TEMP

```
#define DISPLAY_MGT_LINE_TEMP 0
```

Current temperature is displayed on line 0

Definition at line 18 of file DisplayManagement.h.

4.20.2.5 DISPLAY_MGT_PERIOD_TASK_SENSOR

```
#define DISPLAY_MGT_PERIOD_TASK_SENSOR 1500
```

Display is updated every 1.5s

Definition at line 15 of file DisplayManagement.h.

4.20.2.6 DISPLAY_MGT_PERIOD_WELCOME_MSG_REMOVAL

```
#define DISPLAY_MGT_PERIOD_WELCOME_MSG_REMOVAL 5000
```

Time after which one the welcome message is removed

Definition at line 16 of file DisplayManagement.h.

4.20.3 Variable Documentation

4.20.3.1 humidityDisplayString

```
const uint8_t humidityDisplayString[] = "Humidite :  "
```

String used for humidity display

Definition at line 43 of file DisplayManagement.h.

4.20.3.2 LCD_init_cnf

```
const T_LCD_conf_struct LCD_init_cnf
```

Initial value:

```
= {
    DISPLAY_MGT_I2C_BITRATE,
    DISPLAY_MGT_LCD_I2C_ADDR,
    LCD_CNF_BACKLIGHT_ON,
    LCD_CNF_TWO_LINE,
    LCD_CNF_FONT_5_8,
    LCD_CNF_DISPLAY_ON,
    LCD_CNF_CURSOR_OFF,
    LCD_CNF_CURSOR_BLINK_OFF,
    LCD_CNF_ENTRY_MODE_DIRECTION_RIGHT,
    LCD_CNF_ENTRY_MODE_DISPLAY_SHIFT_OFF
}
```

LCD configuration structure.

This structure defines the initial configuration of the LCD screen.

Definition at line 27 of file DisplayManagement.h.

4.20.3.3 tempDisplayString

```
const uint8_t tempDisplayString[ ] = "Temperature :  "
```

String used for temperature display

Definition at line 42 of file DisplayManagement.h.

4.20.3.4 welcomeMessageString

```
const uint8_t welcomeMessageString[ ] = "Bienvenue !"
```

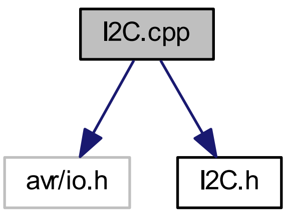
Definition at line 41 of file DisplayManagement.h.

4.21 I2C.cpp File Reference

Two-wire interface (I2C) source file.

```
#include <avr/io.h>
#include "I2C.h"
```

Include dependency graph for I2C.cpp:



4.21.1 Detailed Description

Two-wire interface (I2C) source file.

Date

19 avr. 2019

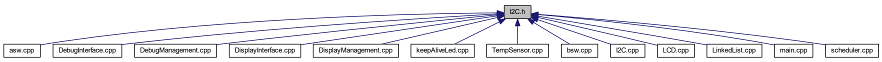
Author

nicls67

4.22 I2C.h File Reference

I2C class header file.

This graph shows which files directly or indirectly include this file:



Classes

- class I2C

  Two-wire serial interface (I2C) class definition.

Macros

- #define START 0x08
- #define SLA_ACK 0x18
- #define DATA_ACK 0x28

4.22.1 Detailed Description

I2C class header file.

Date

19 avr. 2019

Author

nicls67

4.22.2 Macro Definition Documentation

4.22.2.1 DATA_ACK

```
#define DATA_ACK 0x28
```

TWSR status code : DATA has been transmitted and ACK has been received

Definition at line 15 of file I2C.h.

4.22.2.2 SLA_ACK

```
#define SLA_ACK 0x18
```

TWSR status code : SLA has been transmitted and ACK has been received

Definition at line 14 of file I2C.h.

4.22.2.3 START

```
#define START 0x08
```

TWSR status code : START condition transmitted

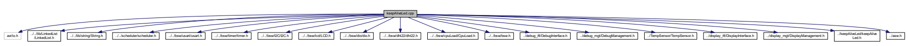
Definition at line 13 of file I2C.h.

4.23 keepAliveLed.cpp File Reference

Definition of function for class keepAliveLed.

```
#include <avr/io.h>
#include "../../lib/LinkedList/LinkedList.h"
#include "../../lib/string/String.h"
#include "../../scheduler/scheduler.h"
#include "../../bsw/usart/usart.h"
#include "../../bsw/timer/timer.h"
#include "../../bsw/I2C/I2C.h"
#include "../../bsw/lcd/LCD.h"
#include "../../bsw/dio/dio.h"
#include "../../bsw/dht22/dht22.h"
#include "../../bsw/cpuLoad/CpuLoad.h"
#include "../../bsw/bsw.h"
#include "../debug_ift/DebugInterface.h"
#include "../debug_mgt/DebugManagement.h"
#include "../TempSensor/TempSensor.h"
#include "../display_ift/DisplayInterface.h"
#include "../display_mgt/DisplayManagement.h"
#include "../keepAliveLed/keepAliveLed.h"
#include "../asw.h"
```

Include dependency graph for keepAliveLed.cpp:



4.23.1 Detailed Description

Definition of function for class keepAliveLed.

Date

17 mars 2018

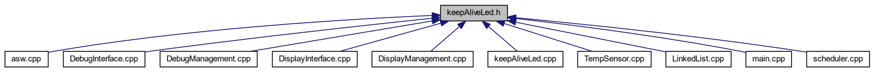
Author

nicls67

4.24 keepAliveLed.h File Reference

Class keepAliveLed header file.

This graph shows which files directly or indirectly include this file:



Classes

- class keepAliveLed

    Class for keep-alive LED blinking.

Macros

- #define PERIOD_MS_TASK_LED SW_PERIOD_MS
- #define LED_PORT ENCODE_PORT(PORT_B, 7)

4.24.1 Detailed Description

Class keepAliveLed header file.

Date

17 mars 2018

Author

nicls67

4.24.2 Macro Definition Documentation

4.24.2.1 LED_PORT

```
#define LED_PORT ENCODE_PORT(PORT_B, 7)
```

LED is connected to port PB7

Definition at line 16 of file keepAliveLed.h.

4.24.2.2 PERIOD_MS_TASK_LED

```
#define PERIOD_MS_TASK_LED SW_PERIOD_MS
```

Period for led blinking

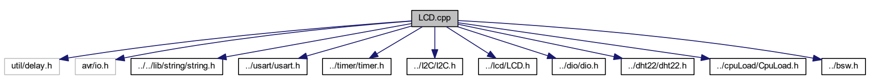
Definition at line 15 of file keepAliveLed.h.

4.25 LCD.cpp File Reference

LCD class source file.

```
#include <util/delay.h>
#include <avr/io.h>
#include "../../lib/string/string.h"
#include "../usart/usart.h"
#include "../timer/timer.h"
#include "../I2C/I2C.h"
#include "../lcd/LCD.h"
#include "../dio/dio.h"
#include "../dht22/dht22.h"
#include "../cpuLoad/CpuLoad.h"
#include "../bsw.h"
```

Include dependency graph for LCD.cpp:



4.25.1 Detailed Description

LCD class source file.

Date

20 avr. 2019

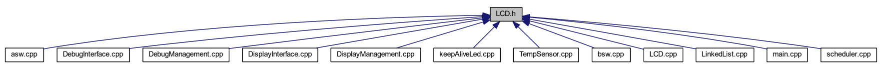
Author

nicls67

4.26 LCD.h File Reference

LCD class header file.

This graph shows which files directly or indirectly include this file:



Classes

- struct T_LCD_conf_struct

    Structure defining LCD configuration.
- class LCD

    Class for LCD S2004A display driver.

Macros

- #define EN_PIN 2
- #define RW_PIN 1
- #define RS_PIN 0
- #define BACKLIGHT_PIN 3
- #define LCD_INST_CLR_DISPLAY_BIT 0
- #define LCD_INST_FUNCTION_SET 5
- #define LCD_INST_DISPLAY_CTRL 3
- #define LCD_INST_ENTRY_MODE_SET 2
- #define LCD_INST_SET_DDRAM_ADDR 7
- #define LCD_FCT_SET_FIELD_DL 4
- #define LCD_FCT_SET_FIELD_N 3
- #define LCD_FCT_SET_FIELD_F 2
- #define LCD_DISPLAY_CTRL_FIELD_D 2
- #define LCD_DISPLAY_CTRL_FIELD_C 1
- #define LCD_DISPLAY_CTRL_FIELD_B 0
- #define LCD_CNF_SHIFT_ID 1
- #define LCD_CNF_SHIFT_SH 0
- #define LCD_CNF_ONE_LINE 0

- #define LCD_CNF_TWO_LINE 1
- #define LCD_CNF_FONT_5_8 0
- #define LCD_CNF_FONT_5_11 1
- #define LCD_CNF_DISPLAY_ON 1
- #define LCD_CNF_DISPLAY_OFF 0
- #define LCD_CNF_CURSOR_ON 1
- #define LCD_CNF_CURSOR_OFF 0
- #define LCD_CNF_CURSOR_BLINK_ON 1
- #define LCD_CNF_CURSOR_BLINK_OFF 0
- #define LCD_CNF_ENTRY_MODE_DIRECTION_RIGHT 1
- #define LCD_CNF_ENTRY_MODE_DIRECTION_LEFT 0
- #define LCD_CNF_ENTRY_MODE_DISPLAY_SHIFT_ON 1
- #define LCD_CNF_ENTRY_MODE_DISPLAY_SHIFT_OFF 0
- #define LCD_CNF_BACKLIGHT_ON 1
- #define LCD_CNF_BACKLIGHT_OFF 0
- #define LCD_RAM_1_LINE_MIN 0
- #define LCD_RAM_1_LINE_MAX 0x4F
- #define LCD_RAM_2_LINES_MIN_1 0
- #define LCD_RAM_2_LINES_MAX_1 0x27
- #define LCD_RAM_2_LINES_MIN_2 0x40
- #define LCD_RAM_2_LINES_MAX_2 0x67
- #define LCD_WAIT_CLR_RETURN 1600
- #define LCD_WAIT_OTHER_MODES 40
- #define LCD_SIZE_NB_CHAR_PER_LINE 20
- #define LCD_SIZE_NB_LINES 4

Enumerations

- enum T_LCD_command {
  LCD_CMD_FUNCTION_SET, LCD_CMD_CLEAR_DISPLAY, LCD_CMD_DISPLAY_CTRL, LCD_CMD_ENTRY_MODE_SET,
  LCD_CMD_SET_DDRAM_ADDR }

  LCD commands enumeration.
- enum T_LCD_config_mode { LCD_MODE_INSTRUCTION = 0, LCD_MODE_DATA = 1 }

  LCD modes enumeration.
- enum T_LCD_ram_area { LCD_DATA_DDRAM, LCD_DATA_CGRAM }

  Screen RAM definition.

4.26.1 Detailed Description

LCD class header file.

Date

20 avr. 2019

Author

nicls67

4.26.2 Macro Definition Documentation

4.26.2.1 BACKLIGHT_PIN

```
#define BACKLIGHT_PIN 3
```

Backlight pin is on P3

Definition at line 17 of file LCD.h.

4.26.2.2 EN_PIN

```
#define EN_PIN 2
```

EN bit is on P2

Definition at line 14 of file LCD.h.

4.26.2.3 LCD_CNF_BACKLIGHT_OFF

```
#define LCD_CNF_BACKLIGHT_OFF 0
```

Backlight is disabled

Definition at line 70 of file LCD.h.

4.26.2.4 LCD_CNF_BACKLIGHT_ON

```
#define LCD_CNF_BACKLIGHT_ON 1
```

Backlight is enabled

Definition at line 69 of file LCD.h.

4.26.2.5 LCD_CNF_CURSOR_BLINK_OFF

```
#define LCD_CNF_CURSOR_BLINK_OFF 0
```

Cursor blinking is off, bit is set to 0

Definition at line 58 of file LCD.h.

4.26.2.6 LCD_CNF_CURSOR_BLINK_ON

```
#define LCD_CNF_CURSOR_BLINK_ON 1
```

Cursor blinking is on, bit is set to 1

Definition at line 57 of file LCD.h.

4.26.2.7 LCD_CNF_CURSOR_OFF

```
#define LCD_CNF_CURSOR_OFF 0
```

Cursor is off, bit is set to 0

Definition at line 54 of file LCD.h.

4.26.2.8 LCD_CNF_CURSOR_ON

```
#define LCD_CNF_CURSOR_ON 1
```

Cursor is on, bit is set to 1

Definition at line 53 of file LCD.h.

4.26.2.9 LCD_CNF_DISPLAY_OFF

```
#define LCD_CNF_DISPLAY_OFF 0
```

Display is off, bit is set to 0

Definition at line 50 of file LCD.h.

4.26.2.10 LCD_CNF_DISPLAY_ON

```
#define LCD_CNF_DISPLAY_ON 1
```

Display is on, bit is set to 1

Definition at line 49 of file LCD.h.

4.26.2.11 LCD_CNF_ENTRY_MODE_DIRECTION_LEFT

```
#define LCD_CNF_ENTRY_MODE_DIRECTION_LEFT 0
```

Direction of shift is left, bit is set to 0

Definition at line 62 of file LCD.h.

4.26.2.12 LCD_CNF_ENTRY_MODE_DIRECTION_RIGHT

```
#define LCD_CNF_ENTRY_MODE_DIRECTION_RIGHT 1
```

Direction of shift is right, bit is set to 1

Definition at line 61 of file LCD.h.

4.26.2.13 LCD_CNF_ENTRY_MODE_DISPLAY_SHIFT_OFF

```
#define LCD_CNF_ENTRY_MODE_DISPLAY_SHIFT_OFF 0
```

Display shift is not performed, bit is set to 0

Definition at line 66 of file LCD.h.

4.26.2.14 LCD_CNF_ENTRY_MODE_DISPLAY_SHIFT_ON

```
#define LCD_CNF_ENTRY_MODE_DISPLAY_SHIFT_ON 1
```

Display shift is performed, bit is set to 1

Definition at line 65 of file LCD.h.

4.26.2.15 LCD_CNF_FONT_5_11

```
#define LCD_CNF_FONT_5_11 1
```

Two-line configuration, bit is set to 1

Definition at line 46 of file LCD.h.

4.26.2.16 LCD_CNF_FONT_5_8

```
#define LCD_CNF_FONT_5_8 0
```

One-line configuration, bit is set to 0

Definition at line 45 of file LCD.h.

4.26.2.17 LCD_CNF_ONE_LINE

```
#define LCD_CNF_ONE_LINE 0
```

One-line configuration, bit is set to 0

Definition at line 41 of file LCD.h.

4.26.2.18 LCD_CNF_SHIFT_ID

```
#define LCD_CNF_SHIFT_ID 1
```

Field ID (increment or decrement) of command "entry mode set" is on bit DB1

Definition at line 37 of file LCD.h.

4.26.2.19 LCD_CNF_SHIFT_SH

```
#define LCD_CNF_SHIFT_SH 0
```

Field SH (shift of display) of command "entry mode set" is on bit DB1

Definition at line 38 of file LCD.h.

4.26.2.20 LCD_CNF_TWO_LINE

```
#define LCD_CNF_TWO_LINE 1
```

Two-line configuration, bit is set to 1

Definition at line 42 of file LCD.h.

4.26.2.21 LCD_DISPLAY_CTRL_FIELD_B

```
#define LCD_DISPLAY_CTRL_FIELD_B 0
```

Field B (cursor blink) of command "display control" is on bit DB0

Definition at line 34 of file LCD.h.

4.26.2.22 LCD_DISPLAY_CTRL_FIELD_C

```
#define LCD_DISPLAY_CTRL_FIELD_C 1
```

Field C (cursor on/off) of command "display control" is on bit DB1

Definition at line 33 of file LCD.h.

4.26.2.23 LCD_DISPLAY_CTRL_FIELD_D

```
#define LCD_DISPLAY_CTRL_FIELD_D 2
```

Field D (display on/off) of command "display control" is on bit DB2

Definition at line 32 of file LCD.h.

4.26.2.24 LCD_FCT_SET_FIELD_DL

```
#define LCD_FCT_SET_FIELD_DL 4
```

Field DL (data length) of command "function set" is on bit DB4

Definition at line 27 of file LCD.h.

4.26.2.25 LCD_FCT_SET_FIELD_F

```
#define LCD_FCT_SET_FIELD_F 2
```

Field F (font type) of command "function set" is on bit DB2

Definition at line 29 of file LCD.h.

4.26.2.26 LCD_FCT_SET_FIELD_N

```
#define LCD_FCT_SET_FIELD_N 3
```

Field N (number of lines) of command "function set" is on bit DB3

Definition at line 28 of file LCD.h.

4.26.2.27 LCD_INST_CLR_DISPLAY_BIT

```
#define LCD_INST_CLR_DISPLAY_BIT 0
```

Instruction bit for "clear display" is DB0

Definition at line 20 of file LCD.h.

4.26.2.28 LCD_INST_DISPLAY_CTRL

```
#define LCD_INST_DISPLAY_CTRL 3
```

Instruction bit for "display control" is DB3

Definition at line 22 of file LCD.h.

4.26.2.29 LCD_INST_ENTRY_MODE_SET

```
#define LCD_INST_ENTRY_MODE_SET 2
```

Instruction bit for "entry mode" is DB2

Definition at line 23 of file LCD.h.

4.26.2.30 LCD_INST_FUNCTION_SET

```
#define LCD_INST_FUNCTION_SET 5
```

Instruction bit for "function set" is DB5

Definition at line 21 of file LCD.h.

4.26.2.31 LCD_INST_SET_DDRAM_ADDR

```
#define LCD_INST_SET_DDRAM_ADDR 7
```

Instruction bit for "set DDRAM address" is DB7

Definition at line 24 of file LCD.h.

4.26.2.32 LCD_RAM_1_LINE_MAX

```
#define LCD_RAM_1_LINE_MAX 0x4F
```

Maximum address value in 1-line mode

Definition at line 74 of file LCD.h.

4.26.2.33 LCD_RAM_1_LINE_MIN

```
#define LCD_RAM_1_LINE_MIN 0
```

Minimum address value in 1-line mode

Definition at line 73 of file LCD.h.

4.26.2.34 LCD_RAM_2_LINES_MAX_1

```
#define LCD_RAM_2_LINES_MAX_1 0x27
```

Maximum address value in 2-lines mode for line 1

Definition at line 76 of file LCD.h.

4.26.2.35 LCD_RAM_2_LINES_MAX_2

```
#define LCD_RAM_2_LINES_MAX_2 0x67
```

Maximum address value in 2-lines mode for line 2

Definition at line 78 of file LCD.h.

4.26.2.36 LCD_RAM_2_LINES_MIN_1

```
#define LCD_RAM_2_LINES_MIN_1 0
```

Minimum address value in 2-lines mode for line 1

Definition at line 75 of file LCD.h.

4.26.2.37 LCD_RAM_2_LINES_MIN_2

```
#define LCD_RAM_2_LINES_MIN_2 0x40
```

Minimum address value in 2-lines mode for line 2

Definition at line 77 of file LCD.h.

4.26.2.38 LCD_SIZE_NB_CHAR_PER_LINE

```
#define LCD_SIZE_NB_CHAR_PER_LINE 20
```

LCD screen has 20 characters per line

Definition at line 85 of file LCD.h.

4.26.2.39 LCD_SIZE_NB_LINES

```
#define LCD_SIZE_NB_LINES 4
```

LCD screen has 4 lines

Definition at line 86 of file LCD.h.

4.26.2.40 LCD_WAIT_CLR_RETURN

```
#define LCD_WAIT_CLR_RETURN 1600
```

Waiting time after clear display and return home operations is at least 1520 us

Definition at line 81 of file LCD.h.

4.26.2.41 LCD_WAIT_OTHER_MODES

```
#define LCD_WAIT_OTHER_MODES 40
```

Waiting time after all other modes is at least 38 us

Definition at line 82 of file LCD.h.

4.26.2.42 RS_PIN

```
#define RS_PIN 0
```

RS pin is on P0

Definition at line 16 of file LCD.h.

4.26.2.43 RW_PIN

```
#define RW_PIN 1
```

RW pin is on P1

Definition at line 15 of file LCD.h.

4.26.3 Enumeration Type Documentation

4.26.3.1 T_LCD_command

```
enum T_LCD_command
```

LCD commands enumeration.

This enumeration defines all command modes available for LCD configuration

Enumerator

LCD_CMD_FUNCTION_SET	
LCD_CMD_CLEAR_DISPLAY	
LCD_CMD_DISPLAY_CTRL	
LCD_CMD_ENTRY_MODE_SET	
LCD_CMD_SET_DDRAM_ADDR	

Definition at line 93 of file LCD.h.

4.26.3.2 T_LCD_config_mode

`enum T_LCD_config_mode`

LCD modes enumeration.

This enumeration defines the possible modes for communication with LCD. Two modes are possible, DATA for writing data in RAM and INSTRUCTION for configuring the display

Enumerator

LCD_MODE_INSTRUCTION	
LCD_MODE_DATA	

Definition at line 107 of file LCD.h.

4.26.3.3 T_LCD_ram_area

`enum T_LCD_ram_area`

Screen RAM definition.

This enumeration defines the RAM areas of the LCD screen : DDRAM for display, CGRAM for characters generation

Enumerator

LCD_DATA_DDRAM	
LCD_DATA_CGRAM	

Definition at line 118 of file LCD.h.

4.27 LinkedList.cpp File Reference

```
#include <stdlib.h>
#include <avr/io.h>
#include "../string/String.h"
#include "../operators/operators.h"
#include "../../bsw/usart/usart.h"
#include "../../bsw/timer/timer.h"
#include "../../bsw/I2C/I2C.h"
#include "../../bsw/lcd/LCD.h"
#include "../../bsw/dio/dio.h"
```

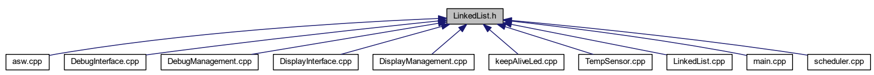
```
#include "../../bsw/dht22/dht22.h"
#include "../../bsw/cpuLoad/CpuLoad.h"
#include "../../bsw/bsw.h"
#include "../../asw/debug_ift/DebugInterface.h"
#include "../../asw/debug_mgt/DebugManagement.h"
#include "../../asw/display_ift/DisplayInterface.h"
#include "../../asw/keepAliveLed/keepAliveLed.h"
#include "../../asw/TempSensor/TempSensor.h"
#include "../../asw/display_mgt/DisplayManagement.h"
#include "../../asw/asw.h"
#include "LinkedList.h"
```

Include dependency graph for LinkedList.cpp:

4.28 LinkedList.h File Reference

Linked List library header file.

This graph shows which files directly or indirectly include this file:



Classes

- class LinkedList

  Linked list class.
- struct LinkedList::T_LL_element

  Type defining a linked list element.

Typedefs

- typedef bool(∗ CompareFctPtr_t) (void ∗LLElement, void ∗CompareElement)

4.28.1 Detailed Description

Linked List library header file.

Date

27 avr. 2019

Author

nicls67

4.28.2 Typedef Documentation

4.28.2.1 CompareFctPtr_t

```
typedef bool(* CompareFctPtr_t) (void *LLElement, void *CompareElement)
```

Type defining a pointer to the comparison function

Definition at line 14 of file LinkedList.h.

4.29 main.cpp File Reference

Background task file.

```
#include <stdlib.h>
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "lib/LinkedList/LinkedList.h"
#include "lib/string/String.h"
#include "scheduler/scheduler.h"
#include "bsw/usart/usart.h"
#include "bsw/timer/timer.h"
#include "bsw/I2C/I2C.h"
#include "bsw/lcd/LCD.h"
#include "bsw/dio/dio.h"
#include "bsw/dht22/dht22.h"
#include "bsw/cpuLoad/CpuLoad.h"
#include "bsw/bsw.h"
#include "asw/debug_ift/DebugInterface.h"
#include "asw/debug_mgt/DebugManagement.h"
#include "asw/TempSensor/TempSensor.h"
#include "asw/display_ift/DisplayInterface.h"
#include "asw/display_mgt/DisplayManagement.h"
#include "asw/keepAliveLed/keepAliveLed.h"
#include "asw/asw.h"
#include "main.h"
```

Include dependency graph for main.cpp:

Functions

- ISR (TIMER1_COMPA_vect)

    Main software interrupt.
- ISR (USART0_RX_vect)

    USART Rx Complete interrupt.
- int main (void)

    Background task of program.

4.29.1 Detailed Description

Background task file.

Date

12 mars 2018

Author

nicls67

4.29.2 Function Documentation

4.29.2.1 ISR() [1/2]

```
ISR (
        TIMER1_COMPA_vect )
```

Main software interrupt.

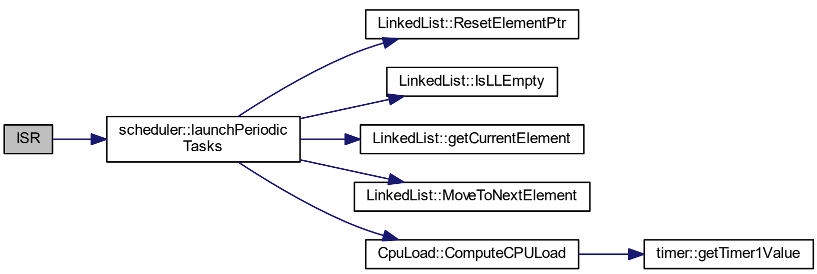
This function handles the interrupt raised by Timer #1. It wakes up the software every 500 ms to perform applications.

Returns

Nothing

Definition at line 46 of file main.cpp.

Here is the call graph for this function:



4.29.2.2 ISR() [2/2]

```
ISR (
            USART0_RX_vect  )
```

USART Rx Complete interrupt.

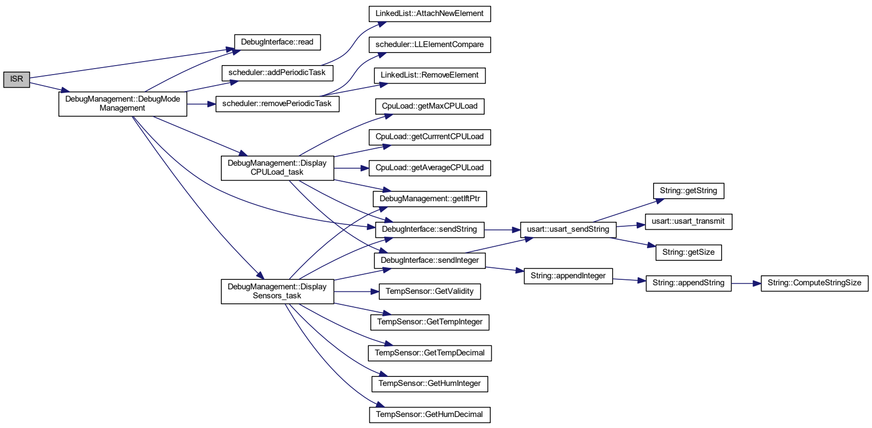
This function handles the interrupt raised when a frame has been received by USART. If debug mode mode is active, it calls debug mode management function. If inactive, it calls debug mode activation function if the received character is 'a'

Returns

Nothing

Definition at line 58 of file main.cpp.

Here is the call graph for this function:



4.29.2.3 main()

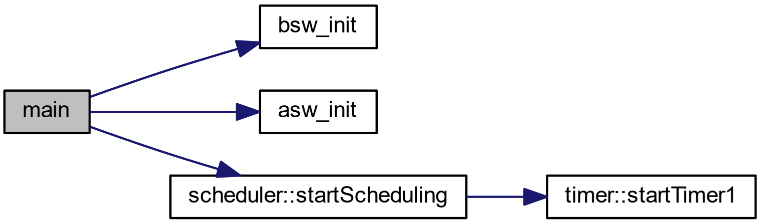
```
int main (
            void  )
```

Background task of program.

This function initializes all the software and then goes into an infinite loop. Periodic interrupt will wake up the software to perform application

Definition at line 84 of file main.cpp.

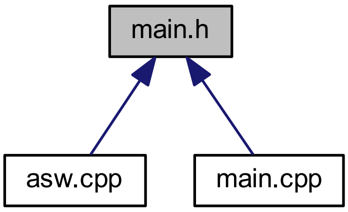
Here is the call graph for this function:



4.30 main.h File Reference

Background task header file.

This graph shows which files directly or indirectly include this file:



Macros

- #define DEBUG_MODE

4.30.1 Detailed Description

Background task header file.

Date

17 mars 2018

Author

nicls67

4.30.2 Macro Definition Documentation

4.30.2.1 DEBUG_MODE

```
#define DEBUG_MODE
```

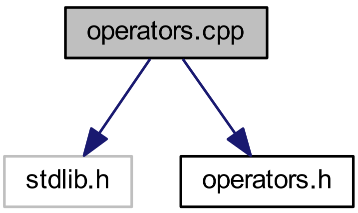
Definition at line 13 of file main.h.

4.31 operators.cpp File Reference

c++ operators definitions

```
#include <stdlib.h>
#include "operators.h"
```

Include dependency graph for operators.cpp:



Functions

- void * operator new (size_t a_size)

    Operator new.
- void operator delete (void *ptr)

    Operator delete.

4.31.1 Detailed Description

c++ operators definitions

Date

14 mars 2018

Author

nicls67

4.31.2 Function Documentation

4.31.2.1 operator delete()

```
void operator delete (
            void * ptr )
```

Operator delete.

Equivalent to free function in C Free the memory zone at address ptr

Parameters

in	*ptr*	Pointer to the start of memory zone to free

Returns

Nothing

Definition at line 18 of file operators.cpp.

4.31.2.2 operator new()

```
void* operator new (
            size_t a_size )
```

Operator new.

Equivalent to malloc function in C Allocates a memory zone of size a_size

Parameters

in	*a_size*	memory size to allocate

Returns

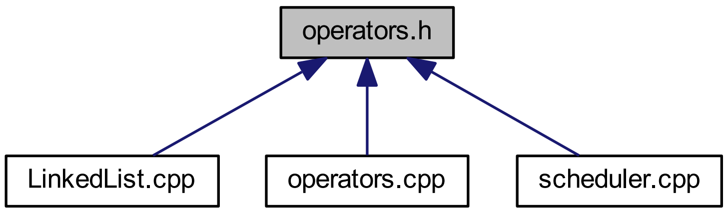
Pointer to the start of allocated memory zone

Definition at line 13 of file operators.cpp.

4.32 operators.h File Reference

c++ operators definitions header file

This graph shows which files directly or indirectly include this file:



Functions

- void ∗ operator new (size_t a_size)

  Operator new.
- void operator delete (void ∗ptr)

  Operator delete.

4.32.1 Detailed Description

c++ operators definitions header file

Date

14 mars 2018

Author

nicls67

4.32.2 Function Documentation

4.32.2.1 operator delete()

```
void operator delete (
            void * ptr )
```

Operator delete.

Equivalent to free function in C Free the memory zone at address ptr

Parameters

in	*ptr*	Pointer to the start of memory zone to free

Returns

Nothing

Definition at line 18 of file operators.cpp.

4.32.2.2 operator new()

```
void* operator new (
            size_t a_size )
```

Operator new.

Equivalent to malloc function in C Allocates a memory zone of size a_size

Parameters

in	*a_size*	memory size to allocate

Returns

Pointer to the start of allocated memory zone

Definition at line 13 of file operators.cpp.

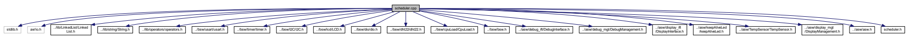
4.33 scheduler.cpp File Reference

Defines scheduler class.

```
#include <stdlib.h>
#include <avr/io.h>
#include "../lib/LinkedList/LinkedList.h"
#include "../lib/string/String.h"
#include "../lib/operators/operators.h"
#include "../bsw/usart/usart.h"
#include "../bsw/timer/timer.h"
#include "../bsw/I2C/I2C.h"
#include "../bsw/lcd/LCD.h"
#include "../bsw/dio/dio.h"
#include "../bsw/dht22/dht22.h"
#include "../bsw/cpuLoad/CpuLoad.h"
#include "../bsw/bsw.h"
#include "../asw/debug_ift/DebugInterface.h"
```

```
#include "../asw/debug_mgt/DebugManagement.h"
#include "../asw/display_ift/DisplayInterface.h"
#include "../asw/keepAliveLed/keepAliveLed.h"
#include "../asw/TempSensor/TempSensor.h"
#include "../asw/display_mgt/DisplayManagement.h"
#include "../asw/asw.h"
#include "scheduler.h"
```

Include dependency graph for scheduler.cpp:



Variables

- scheduler * p_scheduler

4.33.1 Detailed Description

Defines scheduler class.

Date

16 mars 2018

Author

nicls67

4.33.2 Variable Documentation

4.33.2.1 p_scheduler

```
scheduler* p_scheduler
```

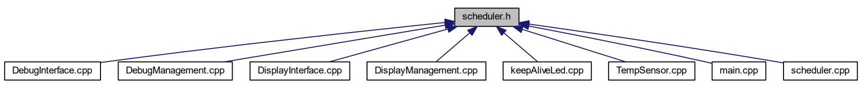
Pointer to scheduler object

Definition at line 38 of file scheduler.cpp.

4.34 scheduler.h File Reference

Scheduler class header file.

This graph shows which files directly or indirectly include this file:



Classes

- class scheduler

  Scheduler class.
- struct scheduler::Task_t

  Type defining a task structure.

Macros

- #define SW_PERIOD_MS 500
- #define PRESCALER_PERIODIC_TIMER 256
- #define TIMER_CTC_VALUE ((F_CPU/PRESCALER_PERIODIC_TIMER)/(1000/SW_PERIOD_MS))

Typedefs

- typedef void(∗ TaskPtr_t) (void)

  Type defining a pointer to function.

Variables

- scheduler ∗ p_scheduler

4.34.1 Detailed Description

Scheduler class header file.

Date

16 mars 2018

Author

nicls67

4.34.2 Macro Definition Documentation

4.34.2.1 PRESCALER_PERIODIC_TIMER

```
#define PRESCALER_PERIODIC_TIMER 256
```

Value of prescaler to use for periodic timer

Definition at line 16 of file scheduler.h.

4.34.2.2 SW_PERIOD_MS

```
#define SW_PERIOD_MS 500
```

Software period, used to define periodic timer interrupt

Definition at line 15 of file scheduler.h.

4.34.2.3 TIMER_CTC_VALUE

```
#define TIMER_CTC_VALUE ((F_CPU/PRESCALER_PERIODIC_TIMER)/(1000/SW_PERIOD_MS))
```

Compare value for periodic timer

Definition at line 17 of file scheduler.h.

4.34.3 Typedef Documentation

4.34.3.1 TaskPtr_t

```
typedef void(* TaskPtr_t) (void)
```

Type defining a pointer to function.

Definition at line 22 of file scheduler.h.

4.34.4 Variable Documentation

4.34.4.1 p_scheduler

```
scheduler* p_scheduler
```

Pointer to scheduler object

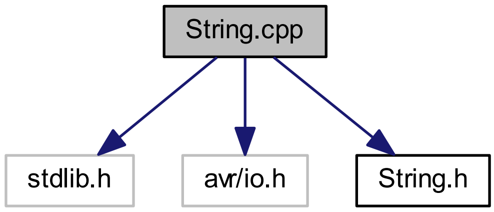
Definition at line 38 of file scheduler.cpp.

4.35 String.cpp File Reference

String class source file.

```
#include <stdlib.h>
#include <avr/io.h>
#include "String.h"
```

Include dependency graph for String.cpp:



4.35.1 Detailed Description

String class source file.

Date

2 mai 2019

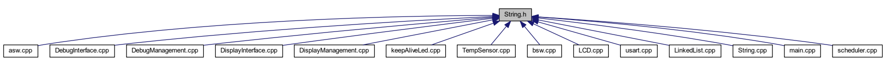
Author

nicls67

4.36 String.h File Reference

String class header file.

This graph shows which files directly or indirectly include this file:



Classes

- class String

  String management class.

4.36.1 Detailed Description

String class header file.

Date

2 mai 2019

Author

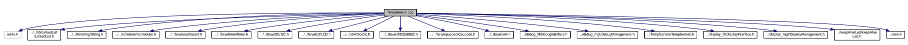
nicls67

4.37 TempSensor.cpp File Reference

Defines function of class TempSensor.

```
#include <avr/io.h>
#include "../../lib/LinkedList/LinkedList.h"
#include "../../lib/string/String.h"
#include "../../scheduler/scheduler.h"
#include "../../bsw/usart/usart.h"
#include "../../bsw/timer/timer.h"
#include "../../bsw/I2C/I2C.h"
#include "../../bsw/lcd/LCD.h"
#include "../../bsw/dio/dio.h"
#include "../../bsw/dht22/dht22.h"
#include "../../bsw/cpuLoad/CpuLoad.h"
#include "../../bsw/bsw.h"
#include "../debug_ift/DebugInterface.h"
#include "../debug_mgt/DebugManagement.h"
#include "../TempSensor/TempSensor.h"
#include "../display_ift/DisplayInterface.h"
#include "../display_mgt/DisplayManagement.h"
#include "../keepAliveLed/keepAliveLed.h"
#include "../asw.h"
```

Include dependency graph for TempSensor.cpp:



Macros

- #define PIT_BEFORE_INVALID 60

4.37.1 Detailed Description

Defines function of class TempSensor.

Date

23 mars 2018

Author

nicls67

4.37.2 Macro Definition Documentation

4.37.2.1 PIT_BEFORE_INVALID

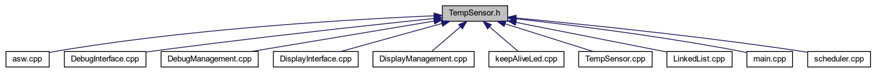
```
#define PIT_BEFORE_INVALID 60
```

Definition at line 35 of file TempSensor.cpp.

4.38 TempSensor.h File Reference

Class TempSensor header file.

This graph shows which files directly or indirectly include this file:



Classes

- class TempSensor

    Class for temperature sensor.

Macros

- #define PERIOD_MS_TASK_TEMP_SENSOR 5000

4.38.1 Detailed Description

Class TempSensor header file.

Date

23 mars 2018

Author

nicls67

4.38.2 Macro Definition Documentation

4.38.2.1 PERIOD_MS_TASK_TEMP_SENSOR

```
#define PERIOD_MS_TASK_TEMP_SENSOR 5000
```

Period for reading temperature data

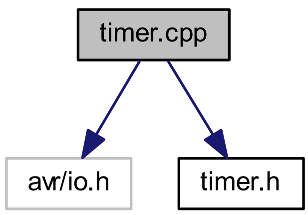
Definition at line 13 of file TempSensor.h.

4.39 timer.cpp File Reference

Defines function for class timer.

```
#include <avr/io.h>
#include "timer.h"
```

Include dependency graph for timer.cpp:



4.39.1 Detailed Description

Defines function for class timer.

Date

15 mars 2018

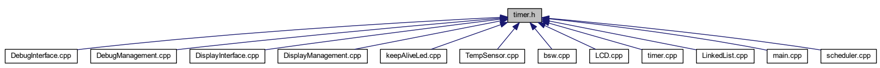
Author

nicls67

4.40 timer.h File Reference

Timer class header file.

This graph shows which files directly or indirectly include this file:



Classes

- class timer

  Class defining a timer.

4.40.1 Detailed Description

Timer class header file.

Date

15 mars 2018

Author

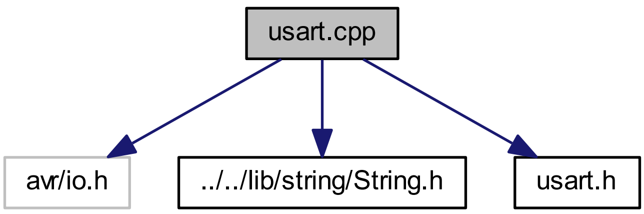
nicls67

4.41 usart.cpp File Reference

BSW library for USART.

```
#include <avr/io.h>
#include "../../lib/string/String.h"
#include "usart.h"
```

Include dependency graph for usart.cpp:



4.41.1 Detailed Description

BSW library for USART.

Date

13 mars 2018

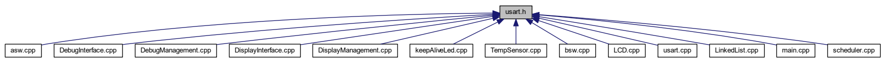
Author

nicls67

4.42 usart.h File Reference

Header file for USART library.

This graph shows which files directly or indirectly include this file:



Classes

- class usart

  USART serial bus class.

4.42.1 Detailed Description

Header file for USART library.

Date

13 mars 2018

Author

nicls67

Index

~LinkedList
LinkedList, 76
~String
String, 91

ASW_cnf_struct
asw.cpp, 134
asw.h, 136
addPeriodicTask
scheduler, 83
alignment
T_display_data, 104
appendBool
String, 92
appendChar
String, 93
appendInteger
String, 94
appendString
String, 95
asw.cpp, 133
ASW_cnf_struct, 134
asw_init, 134
asw.h, 135
ASW_cnf_struct, 136
asw_init, 136
asw_init
asw.cpp, 134
asw.h, 136
AttachNewElement
LinkedList, 77
avg_load
CpuLoad, 8

BACKLIGHT_PIN
LCD.h, 169
BSW_cnf_struct
bsw.cpp, 138
bsw.h, 140
backlight_en
T_LCD_conf_struct, 107
backlight_enable
LCD, 71
BaudRate
usart, 131
bitrate
I2C, 55
blinkLed_task
keepAliveLed, 56
bsw.cpp, 137
BSW_cnf_struct, 138
bsw_init, 137
bsw.h, 138
BSW_cnf_struct, 140
bsw_init, 139
bsw_init
bsw.cpp, 137
bsw.h, 139

Clear
String, 95
ClearFullScreen
DisplayInterface, 36
ClearLine
DisplayInterface, 37
ClearStringInDataStruct
DisplayInterface, 38
cnfCursorBlink
LCD, 72
cnfCursorOnOff
LCD, 72
cnfDisplayOnOff
LCD, 72
cnfEntryModeDir
LCD, 72
cnfEntryModeShift
LCD, 72
cnfFontType
LCD, 73
cnfI2C_addr
LCD, 73
cnfLineNumber
LCD, 73
command
LCD, 60
CompareFctPtr_t
LinkedList.h, 180
ComputeCPULoad
CpuLoad, 6
ComputeStringSize
String, 96
ConfigureBacklight
LCD, 61
ConfigureCursorBlink
LCD, 61
ConfigureCursorOnOff
LCD, 62
ConfigureDisplayOnOff
LCD, 63
ConfigureEntryModeDir

LCD, 63
ConfigureEntryModeShift
LCD, 64
ConfigureFontType
LCD, 65
ConfigureI2CAddr
LCD, 65
ConfigureLineNumber
LCD, 66
configureTimer1
timer, 123
CpuLoad, 5
avg_load, 8
ComputeCPULoad, 6
CpuLoad, 6
current_load, 8
getAverageCPULoad, 6
getCurrrentCPULoad, 7
getMaxCPULoad, 7
last_sum_value, 8
max_load, 9
sample_cnt, 9
sample_idx, 9
sample_mem, 9
CpuLoad.cpp, 140
CpuLoad.h, 141
NB_OF_SAMPLES, 141
curElement_ptr
LinkedList, 81
current_load
CpuLoad, 8
cursor_en
T_LCD_conf_struct, 107
cursorBlink_en
T_LCD_conf_struct, 107

DATA_ACK
I2C.h, 164
DDRA_PTR
dio_reg_atm2560.h, 153
DDRB_PTR
dio_reg_atm2560.h, 153
DDRC_PTR
dio_reg_atm2560.h, 153
DDRD_PTR
dio_reg_atm2560.h, 153
DEBUG_MODE
main.h, 184
DECODE_PIN
dio.h, 149
DECODE_PORT
dio.h, 149
DHT22_PORT
dht22.h, 147
DISPLAY_LINE_SHIFT_PERIOD_MS
DisplayInterface.h, 157
DISPLAY_LINE_SHIFT_TEMPO_TIME
DisplayInterface.h, 157
DISPLAY_MGT_I2C_BITRATE
DisplayManagement.h, 160
DISPLAY_MGT_LCD_I2C_ADDR
DisplayManagement.h, 160
DISPLAY_MGT_LINE_HUM
DisplayManagement.h, 160
DISPLAY_MGT_LINE_TEMP
DisplayManagement.h, 161
DISPLAY_MGT_PERIOD_TASK_SENSOR
DisplayManagement.h, 161
DISPLAY_MGT_PERIOD_WELCOME_MSG_REMO↩
VAL
DisplayManagement.h, 161
data_ptr
LinkedList::T_LL_element, 110
ddram_addr
LCD, 73
debug_ift_ptr
DebugManagement, 22
debug_state
DebugManagement, 22
debug_state_t
DebugManagement.h, 145
DebugInterface, 10
DebugInterface, 11
nextLine, 11
read, 11
sendBool, 12
sendChar, 13
sendInteger, 13
sendString, 14, 15
usart_drv_ptr, 16
DebugInterface.cpp, 142
DebugInterface.h, 142
USART_BAUDRATE, 142
DebugManagement, 17
debug_ift_ptr, 22
debug_state, 22
DebugManagement, 18
DebugModeManagement, 18
DisplayCPULoad_task, 19
DisplaySensors_task, 20
getIftPtr, 21
DebugManagement.cpp, 143
str_debug_main_menu, 144
DebugManagement.h, 144
debug_state_t, 145
PERIOD_MS_TASK_DISPLAY_CPU_LOAD, 145
PERIOD_MS_TASK_DISPLAY_SENSORS, 145
DebugModeManagement
DebugManagement, 18
dht22, 22
dht22, 23
initializeCommunication, 23
read, 24
dht22.cpp, 146
MAX_WAIT_TIME_US, 146
dht22.h, 147
DHT22_PORT, 147

dio, 25
  dio, 26
  dio_changePortPinCnf, 27
  dio_getPort, 27
  dio_getPort_fast, 28
  dio_invertPort, 29
  dio_memorizePINaddress, 30
  dio_setPort, 30
  getDDRxAddress, 31
  getPINxAddress, 32
  getPORTxAddress, 32
  PINx_addr_mem, 34
  PINx_idx_mem, 34
  ports_init, 33
dio.cpp, 148
dio.h, 148
  DECODE_PIN, 149
  DECODE_PORT, 149
  ENCODE_PORT, 150
  PORT_CNF_IN, 150
  PORT_CNF_OUT, 151
  PORT_A, 150
  PORT_B, 150
  PORT_C, 150
  PORT_D, 151
dio_changePortPinCnf
  dio, 27
dio_getPort
  dio, 27
dio_getPort_fast
  dio, 28
dio_invertPort
  dio, 29
dio_memorizePINaddress
  dio, 30
dio_port_cnf.h, 151
  PORTB_CNF_DDRB, 152
  PORTB_CNF_PORTB, 152
dio_reg_atm2560.h, 152
  DDRA_PTR, 153
  DDRB_PTR, 153
  DDRC_PTR, 153
  DDRD_PTR, 153
  PINA_PTR, 153
  PINB_PTR, 153
  PINC_PTR, 154
  PIND_PTR, 154
  PORTA_PTR, 154
  PORTB_PTR, 154
  PORTC_PTR, 154
  PORTD_PTR, 155
dio_setPort
  dio, 30
display_data
  DisplayInterface, 45
display_en
  T_LCD_conf_struct, 107
display_str
  T_display_data, 104
DisplayCPULoad_task
  DebugManagement, 19
DisplayFullLine
  DisplayInterface, 39
DisplayInterface, 34
  ClearFullScreen, 36
  ClearLine, 37
  ClearStringInDataStruct, 38
  display_data, 45
  DisplayFullLine, 39
  DisplayInterface, 36
  dummy, 45
  FindFirstCharAddr, 39
  getDisplayDataPtr, 40
  IsLineEmpty, 41
  isShiftInProgress, 45
  p_lcd, 45
  RefreshLine, 41
  setLineAlignment, 42
  setLineAlignmentAndRefresh, 43
  shiftLine_task, 43
  updateLineAndRefresh, 44
DisplayInterface.cpp, 155
DisplayInterface.h, 156
  DISPLAY_LINE_SHIFT_PERIOD_MS, 157
  DISPLAY_LINE_SHIFT_TEMPO_TIME, 157
  T_DisplayInterface_LineAlignment, 157
  T_DisplayInterface_LineDisplayMode, 158
DisplayManagement, 46
  DisplayManagement, 47
  DisplaySensorData_Task, 48
  GetIftPointer, 49
  GetTempSensorPtr, 49
  p_display_ift, 51
  p_tempSensor, 51
  RemoveWelcomeMessage_Task, 50
DisplayManagement.cpp, 158
DisplayManagement.h, 159
  DISPLAY_MGT_I2C_BITRATE, 160
  DISPLAY_MGT_LCD_I2C_ADDR, 160
  DISPLAY_MGT_LINE_HUM, 160
  DISPLAY_MGT_LINE_TEMP, 161
  DISPLAY_MGT_PERIOD_TASK_SENSOR, 161
  DISPLAY_MGT_PERIOD_WELCOME_MSG_R↩
    EMOVAL, 161
  humidityDisplayString, 161
  LCD_init_cnf, 161
  tempDisplayString, 162
  welcomeMessageString, 162
DisplaySensorData_Task
  DisplayManagement, 48
DisplaySensors_task
  DebugManagement, 20
dummy
  DisplayInterface, 45

EN_PIN
  LCD.h, 169

ENCODE_PORT
dio.h, 150
entryModeDir
T_LCD_conf_struct, 108
entryModeShift
T_LCD_conf_struct, 108

FindElement
LinkedList, 77
FindFirstCharAddr
DisplayInterface, 39
firstElement
LinkedList, 81
fontType_cnf
T_LCD_conf_struct, 108

getAverageCPULoad
CpuLoad, 6
getCurrentElement
LinkedList, 78
getCurrrentCPULoad
CpuLoad, 7
GetDDRAMAddress
LCD, 67
getDDRxAddress
dio, 31
getDisplayDataPtr
DisplayInterface, 40
GetHumDecimal
TempSensor, 113
GetHumInteger
TempSensor, 113
getHumPtr
TempSensor, 114
getHumidity
TempSensor, 113
GetIftPointer
DisplayManagement, 49
getIftPtr
DebugManagement, 21
GetLineNumberCnf
LCD, 67
getMaxCPULoad
CpuLoad, 7
getPINxAddress
dio, 32
getPORTxAddress
dio, 32
getPitNumber
scheduler, 84
getSize
String, 97
getString
String, 97
getTaskPeriod
TempSensor, 114
getTemp
TempSensor, 115
GetTempDecimal
TempSensor, 115
GetTempInteger
TempSensor, 116
getTempPtr
TempSensor, 116
GetTempSensorPtr
DisplayManagement, 49
getTimer1Value
timer, 124
GetValidity
TempSensor, 117

humidityDisplayString
DisplayManagement.h, 161

I2C.cpp, 162
I2C.h, 163
DATA_ACK, 164
SLA_ACK, 164
START, 164
I2C, 51
bitrate, 55
I2C, 52
initializeBus, 53
setBitRate, 53
setTxAddress, 54
tx_address, 55
writeByte, 54
i2c_addr
T_LCD_conf_struct, 108
i2c_bitrate
T_LCD_conf_struct, 108
i2c_drv_ptr
LCD, 73
ISR
main.cpp, 181
initializeBus
I2C, 53
initializeCommunication
dht22, 23
InitializeScreen
LCD, 67
isEmpty
T_display_data, 104
IsLLEmpty
LinkedList, 78
IsLineEmpty
DisplayInterface, 41
isShiftInProgress
DisplayInterface, 45

keepAliveLed, 55
blinkLed_task, 56
keepAliveLed, 56
keepAliveLed.cpp, 164
keepAliveLed.h, 165
LED_PORT, 166
PERIOD_MS_TASK_LED, 166

LCD.cpp, 166
LCD.h, 167
BACKLIGHT_PIN, 169
EN_PIN, 169
LCD_CNF_BACKLIGHT_OFF, 169
LCD_CNF_BACKLIGHT_ON, 169
LCD_CNF_CURSOR_BLINK_OFF, 169
LCD_CNF_CURSOR_BLINK_ON, 169
LCD_CNF_CURSOR_OFF, 170
LCD_CNF_CURSOR_ON, 170
LCD_CNF_DISPLAY_OFF, 170
LCD_CNF_DISPLAY_ON, 170
LCD_CNF_ENTRY_MODE_DIRECTION_LEFT, 170
LCD_CNF_ENTRY_MODE_DIRECTION_RIGHT, 171
LCD_CNF_ENTRY_MODE_DISPLAY_SHIFT_OFF, 171
LCD_CNF_ENTRY_MODE_DISPLAY_SHIFT_ON, 171
LCD_CNF_FONT_5_11, 171
LCD_CNF_FONT_5_8, 171
LCD_CNF_ONE_LINE, 172
LCD_CNF_SHIFT_ID, 172
LCD_CNF_SHIFT_SH, 172
LCD_CNF_TWO_LINE, 172
LCD_DISPLAY_CTRL_FIELD_B, 172
LCD_DISPLAY_CTRL_FIELD_C, 173
LCD_DISPLAY_CTRL_FIELD_D, 173
LCD_FCT_SET_FIELD_DL, 173
LCD_FCT_SET_FIELD_F, 173
LCD_FCT_SET_FIELD_N, 173
LCD_INST_CLR_DISPLAY_BIT, 174
LCD_INST_DISPLAY_CTRL, 174
LCD_INST_ENTRY_MODE_SET, 174
LCD_INST_FUNCTION_SET, 174
LCD_INST_SET_DDRAM_ADDR, 174
LCD_RAM_1_LINE_MAX, 175
LCD_RAM_1_LINE_MIN, 175
LCD_RAM_2_LINES_MAX_1, 175
LCD_RAM_2_LINES_MAX_2, 175
LCD_RAM_2_LINES_MIN_1, 175
LCD_RAM_2_LINES_MIN_2, 176
LCD_SIZE_NB_CHAR_PER_LINE, 176
LCD_SIZE_NB_LINES, 176
LCD_WAIT_CLR_RETURN, 176
LCD_WAIT_OTHER_MODES, 176
RS_PIN, 177
RW_PIN, 177
T_LCD_command, 177
T_LCD_config_mode, 178
T_LCD_ram_area, 178
LCD_CNF_BACKLIGHT_OFF
LCD.h, 169
LCD_CNF_BACKLIGHT_ON
LCD.h, 169
LCD_CNF_CURSOR_BLINK_OFF
LCD.h, 169
LCD_CNF_CURSOR_BLINK_ON
LCD.h, 169
LCD_CNF_CURSOR_OFF
LCD.h, 170
LCD_CNF_CURSOR_ON
LCD.h, 170
LCD_CNF_DISPLAY_OFF
LCD.h, 170
LCD_CNF_DISPLAY_ON
LCD.h, 170
LCD_CNF_ENTRY_MODE_DIRECTION_LEFT
LCD.h, 170
LCD_CNF_ENTRY_MODE_DIRECTION_RIGHT
LCD.h, 171
LCD_CNF_ENTRY_MODE_DISPLAY_SHIFT_OFF
LCD.h, 171
LCD_CNF_ENTRY_MODE_DISPLAY_SHIFT_ON
LCD.h, 171
LCD_CNF_FONT_5_11
LCD.h, 171
LCD_CNF_FONT_5_8
LCD.h, 171
LCD_CNF_ONE_LINE
LCD.h, 172
LCD_CNF_SHIFT_ID
LCD.h, 172
LCD_CNF_SHIFT_SH
LCD.h, 172
LCD_CNF_TWO_LINE
LCD.h, 172
LCD_DISPLAY_CTRL_FIELD_B
LCD.h, 172
LCD_DISPLAY_CTRL_FIELD_C
LCD.h, 173
LCD_DISPLAY_CTRL_FIELD_D
LCD.h, 173
LCD_FCT_SET_FIELD_DL
LCD.h, 173
LCD_FCT_SET_FIELD_F
LCD.h, 173
LCD_FCT_SET_FIELD_N
LCD.h, 173
LCD_INST_CLR_DISPLAY_BIT
LCD.h, 174
LCD_INST_DISPLAY_CTRL
LCD.h, 174
LCD_INST_ENTRY_MODE_SET
LCD.h, 174
LCD_INST_FUNCTION_SET
LCD.h, 174
LCD_INST_SET_DDRAM_ADDR
LCD.h, 174
LCD_RAM_1_LINE_MAX
LCD.h, 175
LCD_RAM_1_LINE_MIN
LCD.h, 175
LCD_RAM_2_LINES_MAX_1
LCD.h, 175

LCD_RAM_2_LINES_MAX_2
LCD.h, 175
LCD_RAM_2_LINES_MIN_1
LCD.h, 175
LCD_RAM_2_LINES_MIN_2
LCD.h, 176
LCD_SIZE_NB_CHAR_PER_LINE
LCD.h, 176
LCD_SIZE_NB_LINES
LCD.h, 176
LCD_WAIT_CLR_RETURN
LCD.h, 176
LCD_WAIT_OTHER_MODES
LCD.h, 176
LCD_init_cnf
DisplayManagement.h, 161
LCD, 57
backlight_enable, 71
cnfCursorBlink, 72
cnfCursorOnOff, 72
cnfDisplayOnOff, 72
cnfEntryModeDir, 72
cnfEntryModeShift, 72
cnfFontType, 73
cnfI2C_addr, 73
cnfLineNumber, 73
command, 60
ConfigureBacklight, 61
ConfigureCursorBlink, 61
ConfigureCursorOnOff, 62
ConfigureDisplayOnOff, 63
ConfigureEntryModeDir, 63
ConfigureEntryModeShift, 64
ConfigureFontType, 65
ConfigureI2CAddr, 65
ConfigureLineNumber, 66
ddram_addr, 73
GetDDRAMAddress, 67
GetLineNumberCnf, 67
i2c_drv_ptr, 73
InitializeScreen, 67
LCD, 59
SetDDRAMAddress, 68
write, 69
write4bits, 70
WriteInRam, 71
LED_PORT
keepAliveLed.h, 166
LLElementCompare
scheduler, 85
last_sum_value
CpuLoad, 8
launchPeriodicTasks
scheduler, 84
lineNumber_cnf
T_LCD_conf_struct, 109
LinkedList, 74
∼LinkedList, 76
AttachNewElement, 77
curElement_ptr, 81
FindElement, 77
firstElement, 81
getCurrentElement, 78
IsLLEmpty, 78
LinkedList, 76
MoveToNextElement, 79
RemoveElement, 79
ResetElementPtr, 80
T_LL_element, 75
LinkedList.cpp, 178
LinkedList.h, 179
CompareFctPtr_t, 180
LinkedList::T_LL_element, 109
data_ptr, 110
nextElement, 110

MAX_WAIT_TIME_US
dht22.cpp, 146
main
main.cpp, 182
main.cpp, 180
ISR, 181
main, 182
main.h, 183
DEBUG_MODE, 184
max_load
CpuLoad, 9
mode
T_display_data, 104
MoveToNextElement
LinkedList, 79

NB_OF_SAMPLES
CpuLoad.h, 141
nextElement
LinkedList::T_LL_element, 110
nextLine
DebugInterface, 11

operator delete
operators.cpp, 185
operators.h, 186
operator new
operators.cpp, 185
operators.h, 187
operators.cpp, 184
operator delete, 185
operator new, 185
operators.h, 185
operator delete, 186
operator new, 187

p_DebugInterface
T_ASW_cnf_struct, 99
p_DebugManagement
T_ASW_cnf_struct, 99
p_DisplayInterface

T_ASW_cnf_struct, 100
p_DisplayManagement
T_ASW_cnf_struct, 100
p_TempSensor
T_ASW_cnf_struct, 100
p_cpuload
T_BSW_cnf_struct, 101
p_dht22
T_BSW_cnf_struct, 101
p_dio
T_BSW_cnf_struct, 102
p_display_ift
DisplayManagement, 51
p_i2c
T_BSW_cnf_struct, 102
p_keepAliveLed
T_ASW_cnf_struct, 100
p_lcd
DisplayInterface, 45
T_BSW_cnf_struct, 102
p_scheduler
scheduler.cpp, 188
scheduler.h, 190
p_tempSensor
DisplayManagement, 51
p_timer
T_BSW_cnf_struct, 102
p_usart
T_BSW_cnf_struct, 102
PERIOD_MS_TASK_DISPLAY_CPU_LOAD
DebugManagement.h, 145
PERIOD_MS_TASK_DISPLAY_SENSORS
DebugManagement.h, 145
PERIOD_MS_TASK_LED
keepAliveLed.h, 166
PERIOD_MS_TASK_TEMP_SENSOR
TempSensor.h, 193
PINA_PTR
dio_reg_atm2560.h, 153
PINB_PTR
dio_reg_atm2560.h, 153
PINC_PTR
dio_reg_atm2560.h, 154
PIND_PTR
dio_reg_atm2560.h, 154
PINx_addr_mem
dio, 34
PINx_idx_mem
dio, 34
PIT_BEFORE_INVALID
TempSensor.cpp, 193
PORT_CNF_IN
dio.h, 150
PORT_CNF_OUT
dio.h, 151
PORT_A
dio.h, 150
PORT_B
dio.h, 150
PORT_C
dio.h, 150
PORT_D
dio.h, 151
PORTA_PTR
dio_reg_atm2560.h, 154
PORTB_CNF_DDRB
dio_port_cnf.h, 152
PORTB_CNF_PORTB
dio_port_cnf.h, 152
PORTB_PTR
dio_reg_atm2560.h, 154
PORTC_PTR
dio_reg_atm2560.h, 154
PORTD_PTR
dio_reg_atm2560.h, 155
PRESCALER_PERIODIC_TIMER
scheduler.h, 189
period
scheduler::Task_t, 110
pit_number
scheduler, 89
ports_init
dio, 33
prescaler
timer, 125

RS_PIN
LCD.h, 177
RW_PIN
LCD.h, 177
read
DebugInterface, 11
dht22, 24
read_humidity
TempSensor, 120
read_temperature
TempSensor, 120
readTempSensor_task
TempSensor, 117
RefreshLine
DisplayInterface, 41
RemoveElement
LinkedList, 79
removePeriodicTask
scheduler, 86
RemoveWelcomeMessage_Task
DisplayManagement, 50
ResetElementPtr
LinkedList, 80

SLA_ACK
I2C.h, 164
START
I2C.h, 164
SW_PERIOD_MS
scheduler.h, 189
sample_cnt

CpuLoad, 9
sample_idx
CpuLoad, 9
sample_mem
CpuLoad, 9
scheduler, 81
addPeriodicTask, 83
getPitNumber, 84
LLElementCompare, 85
launchPeriodicTasks, 84
pit_number, 89
removePeriodicTask, 86
scheduler, 83
startScheduling, 87
Task_t, 83
TasksLL_ptr, 89
updateTaskPeriod, 88
scheduler.cpp, 187
p_scheduler, 188
scheduler.h, 188
p_scheduler, 190
PRESCALER_PERIODIC_TIMER, 189
SW_PERIOD_MS, 189
TIMER_CTC_VALUE, 190
TaskPtr_t, 190
scheduler::Task_t, 110
period, 110
TaskPtr, 111
sendBool
DebugInterface, 12
sendChar
DebugInterface, 13
sendInteger
DebugInterface, 13
sendString
DebugInterface, 14, 15
setBaudRate
usart, 127
setBitRate
I2C, 53
SetDDRAMAddress
LCD, 68
setLineAlignment
DisplayInterface, 42
setLineAlignmentAndRefresh
DisplayInterface, 43
setTxAddress
I2C, 54
setValidity
TempSensor, 118
shift_data
T_display_data, 104
shiftLine_task
DisplayInterface, 43
size
String, 98
startScheduling
scheduler, 87
startTimer1
timer, 124
stopTimer1
timer, 125
str_cur_ptr
T_Display_shift_data, 106
str_debug_main_menu
DebugManagement.cpp, 144
str_ptr
T_Display_shift_data, 106
String, 90
~String, 91
appendBool, 92
appendChar, 93
appendInteger, 94
appendString, 95
Clear, 95
ComputeStringSize, 96
getSize, 97
getString, 97
size, 98
String, 91
string, 98
string
String, 98
String.cpp, 191
String.h, 191

T_ASW_cnf_struct, 99
p_DebugInterface, 99
p_DebugManagement, 99
p_DisplayInterface, 100
p_DisplayManagement, 100
p_TempSensor, 100
p_keepAliveLed, 100
T_BSW_cnf_struct, 101
p_cpuload, 101
p_dht22, 101
p_dio, 102
p_i2c, 102
p_lcd, 102
p_timer, 102
p_usart, 102
T_Display_shift_data, 105
str_cur_ptr, 106
str_ptr, 106
temporization, 106
T_DisplayInterface_LineAlignment
DisplayInterface.h, 157
T_DisplayInterface_LineDisplayMode
DisplayInterface.h, 158
T_LCD_command
LCD.h, 177
T_LCD_conf_struct, 106
backlight_en, 107
cursor_en, 107
cursorBlink_en, 107
display_en, 107
entryModeDir, 108

entryModeShift, 108
fontType_cnf, 108
i2c_addr, 108
i2c_bitrate, 108
lineNumber_cnf, 109
T_LCD_config_mode
LCD.h, 178
T_LCD_ram_area
LCD.h, 178
T_LL_element
LinkedList, 75
T_display_data, 103
alignment, 104
display_str, 104
isEmpty, 104
mode, 104
shift_data, 104
TIMER_CTC_VALUE
scheduler.h, 190
task_period
TempSensor, 121
Task_t
scheduler, 83
TaskPtr
scheduler::Task_t, 111
TaskPtr_t
scheduler.h, 190
TasksLL_ptr
scheduler, 89
tempDisplayString
DisplayManagement.h, 162
TempSensor, 111
GetHumDecimal, 113
GetHumInteger, 113
getHumPtr, 114
getHumidity, 113
getTaskPeriod, 114
getTemp, 115
GetTempDecimal, 115
GetTempInteger, 116
getTempPtr, 116
GetValidity, 117
read_humidity, 120
read_temperature, 120
readTempSensor_task, 117
setValidity, 118
task_period, 121
TempSensor, 112
updateLastValidValues, 119
updateTaskPeriod, 119
valid_hum, 121
valid_pit, 121
valid_temp, 121
validity, 121
validity_last_read, 122
TempSensor.cpp, 192
PIT_BEFORE_INVALID, 193
TempSensor.h, 193
PERIOD_MS_TASK_TEMP_SENSOR, 193
temporization
T_Display_shift_data, 106
timer, 122
configureTimer1, 123
getTimer1Value, 124
prescaler, 125
startTimer1, 124
stopTimer1, 125
timer, 123
timer.cpp, 194
timer.h, 194
tx_address
I2C, 55

USART_BAUDRATE
DebugInterface.h, 142
updateLastValidValues
TempSensor, 119
updateLineAndRefresh
DisplayInterface, 44
updateTaskPeriod
scheduler, 88
TempSensor, 119
usart, 126
BaudRate, 131
setBaudRate, 127
usart, 126
usart_init, 127
usart_read, 128
usart_sendByte, 128
usart_sendString, 129
usart_transmit, 130
usart.cpp, 195
usart.h, 196
usart_drv_ptr
DebugInterface, 16
usart_init
usart, 127
usart_read
usart, 128
usart_sendByte
usart, 128
usart_sendString
usart, 129
usart_transmit
usart, 130

valid_hum
TempSensor, 121
valid_pit
TempSensor, 121
valid_temp
TempSensor, 121
validity
TempSensor, 121
validity_last_read
TempSensor, 122

welcomeMessageString
DisplayManagement.h, 162
write
LCD, 69
write4bits
LCD, 70
writeByte
I2C, 54
WriteInRam
LCD, 71