

```

1  /*! \file WaspFrame.h
2      \brief Library for creating formatted frames
3
4      Copyright (C) 2015 Libelium Comunicaciones Distribuidas S.L.
5      http://www.libelium.com
6
7      This program is free software: you can redistribute it and/or modify
8      it under the terms of the GNU Lesser General Public License as published by
9      the Free Software Foundation, either version 2.1 of the License, or
10     (at your option) any later version.
11
12     This program is distributed in the hope that it will be useful,
13     but WITHOUT ANY WARRANTY; without even the implied warranty of
14     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
15     GNU Lesser General Public License for more details.
16
17     You should have received a copy of the GNU Lesser General Public License
18     along with this program. If not, see <http://www.gnu.org/licenses/>.
19
20     Version:          1.9
21     Design:           David Gascón
22     Implementation:   Yuri Carmona, Javier Siscart, Joaquín Ruiz
23
24  */
25  /* modificacions crades per RUBÉN ALMANSA PER EXIT RESEARCH GROUP a la línia 1069
26     per els valors del WaspFrame
27     que corresponen als frames creats a WaspFrame.cpp*/
28  /*! \def WaspFrame_h
29  */
30  #ifndef WaspFrame_h
31  #define WaspFrame_h
32
33
34
35  /*****
36   * Includes
37   *****/
38
39  #include <string.h>
40  #include <stdint.h>
41  #include <stdlib.h>
42  #include <inttypes.h>
43  #include <WConstants.h>
44  #include "../WaspAES/WaspAES.h"
45
46
47  /*****
48   * Definitions & Declarations
49   *****/
50
51
52
53  /// Gases Board sensor measurements
54
55  /*! \def SENSOR_CO
56      \brief Carbon Monoxide measurement type
57  */
58  /*! \def SENSOR_CO2
59      \brief Carbon Dioxide measurement type
60  */
61  /*! \def SENSOR_O2
62      \brief Oxygen measurement type
63  */
64  /*! \def SENSOR_CH4
65      \brief Methane measurement type
66  */
67  /*! \def SENSOR_LPG
68      \brief Liquefied Petroleum Gases measurement type
69  */
70  /*! \def SENSOR_NH3
71      \brief Ammonia measurement type
72  */

```

```

73  /*! \def SENSOR_AP1
74      \brief Air Pollutans 1 measurement type
75  */
76  /*! \def SENSOR_AP2
77      \brief Air Pollutans 2 measurement type
78  */
79  /*! \def SENSOR_SV
80      \brief Solvent Vapors measurement type
81  */
82  /*! \def SENSOR_NO2
83      \brief Nitrogen Dioxide measurement type
84  */
85  /*! \def SENSOR_O3
86      \brief Ozone measurement type
87  */
88  /*! \def SENSOR_VOC
89      \brief Hydrocarbons measurement type
90  */
91  /*! \def SENSOR_TC
92      \brief Temperature Celsius measurement type
93  */
94  /*! \def SENSOR_TF
95      \brief Temperature Fahrenheit measurement type
96  */
97  /*! \def SENSOR_HUM
98      \brief Humidity measurement type
99  */
100 /*! \def SENSOR_PA
101     \brief Pressure atmospheric measurement type
102 */
103
104
105 /// Events Board Sensor measurements
106
107 /*! \def SENSOR_PW
108     \brief Pressure/Weight measurement type
109 */
110 /*! \def SENSOR_BEND
111     \brief Bend measurement type
112 */
113 /*! \def SENSOR_VBR
114     \brief Vibration measurement type
115 */
116 /*! \def SENSOR_HALL
117     \brief Hall Effect measurement type
118 */
119 /*! \def SENSOR_LP
120     \brief Liquid Presence measurement type
121 */
122 /*! \def SENSOR_LL
123     \brief Liquid Level measurement type
124 */
125 /*! \def SENSOR_LUM
126     \brief Luminosity measurement type
127 */
128 /*! \def SENSOR_PIR
129     \brief Presence measurement type
130 */
131 /*! \def SENSOR_ST
132     \brief Stretch measurement type
133 */
134
135
136 /// Smart cities sensor measurements
137
138 /*! \def SENSOR_MCP
139     \brief Microphone measurement type
140 */
141 /*! \def SENSOR_CDG
142     \brief Crack detection gauge measurement type
143 */
144 /*! \def SENSOR_CPG
145     \brief Crack propagation gauge measurement type

```

```
146 */
147 /*! \def SENSOR_LD
148 \brief Linear Displacement measurement type
149 */
150 /*! \def SENSOR_DUST
151 \brief Dust measurement type
152 */
153 /*! \def SENSOR_US
154 \brief Ultrasound measurement type
155 */
156
157
158 /// Smart parking sensor measurements
159
160 /*! \def SENSOR_MF
161 \brief Magnetic Field measurement type
162 */
163 /*! \def SENSOR_PS
164 \brief Parking Spot Status measurement type
165 */
166
167 /// Agriculture sensor measurements
168
169 /*! \def SENSOR_AIR
170 \brief Air Temperature / Humidity measurement type
171 */
172 /*! \def SENSOR_SOIL
173 \brief Soil Temperature / Moisture measurement type
174 */
175 /*! \def SENSOR_LW
176 \brief Leaf Wetness measurement type
177 */
178 /*! \def SENSOR_PAR
179 \brief Solar Radiation measurement type
180 */
181 /*! \def SENSOR_UV
182 \brief Ultraviolet Radiation measurement type
183 */
184 /*! \def SENSOR_TD
185 \brief Trunk Diameter measurement type
186 */
187 /*! \def SENSOR_SD
188 \brief Stem Diameter measurement type
189 */
190 /*! \def SENSOR_FD
191 \brief Fruit Diameter measurement type
192 */
193 /*! \def SENSOR_ANE
194 \brief Anemometer measurement type
195 */
196 /*! \def SENSOR_WV
197 \brief Wind Vane measurement type
198 */
199 /*! \def SENSOR_PLV
200 \brief Pluviometer measurement type
201 */
202
203
204 /// Radiation sensor measurements
205
206 /*! \def SENSOR_RAD
207 \brief Geiger tube measurement type
208 */
209
210
211 /// Smart Metering sensor measurements
212
213 /*! \def SENSOR_CU
214 \brief Current measurement type
215 */
216 /*! \def SENSOR_WF
217 \brief Water flow measurement type
218 */
```

```
219  /*! \def SENSOR_LC
220      \brief Load cell measurement type
221  */
222  /*! \def SENSOR_DF
223      \brief Distance foil measurement type
224  */
225
226
227  /// Additional sensor measurements
228
229  /*! \def SENSOR_BAT
230      \brief Battery measurement type
231  */
232  /*! \def SENSOR_GPS
233      \brief Global Positioning System measurement type
234  */
235  /*! \def SENSOR_RSSI
236      \brief RSSI measurement type
237  */
238  /*! \def SENSOR_MAC
239      \brief MAC Address measurement type
240  */
241  /*! \def SENSOR_NA
242      \brief Network Address measurement type
243  */
244  /*! \def SENSOR_NID
245      \brief Network Identifier origin measurement type
246  */
247  /*! \def SENSOR_DATE
248      \brief Date measurement type
249  */
250  /*! \def SENSOR_TIME
251      \brief Time measurement type
252  */
253  /*! \def SENSOR_GMT
254      \brief GMT measurement type
255  */
256  /*! \def SENSOR_RAM
257      \brief RAM measurement type
258  */
259  /*! \def SENSOR_IN_TEMP
260      \brief Internal temperature measurement type
261  */
262  /*! \def SENSOR_MILLIS
263      \brief Millis measurement type
264  */
265
266  /// Special sensor measurements
267
268  /*! \def SENSOR_STR
269      \brief String type
270  */
271
272  /// Smart Water
273
274  /*! \def SENSOR_PH
275      \brief pH measurement type
276  */
277  /*! \def SENSOR_ORP
278      \brief Oxidation Reduction Potential measurement type
279  */
280  /*! \def SENSOR_DI
281      \brief Dissolved Ion measurement type
282  */
283  /*! \def SENSOR_DO
284      \brief Dissolved Oxygen measurement type
285  */
286  /*! \def SENSOR_COND
287      \brief Conductivity measurement type
288  */
289  /*! \def SENSOR_WT
290      \brief Water Temperature measurement type
291  */
```

```

292
293
294 /// Smart Libelium
295 /*! \def SENSOR_DM_ST
296 \brief XBee Digimesh awake time for cyclic sleep mode
297 */
298 /*! \def SENSOR_DM_SP
299 \brief XBee Digimesh asleep time for cyclic sleep mode
300 */
301 /*! \def SENSOR_TX_PWR
302 \brief XBee transmission power
303 */
304 /*! \def SENSOR_LUX
305 \brief Luxes measurement type
306 */
307
308
309 /// GPS
310 /*! \def SENSOR_SPEED
311 \brief GPS speed over the ground measurement type
312 */
313 /*! \def SENSOR_COURSE
314 \brief GPS course over the ground measurement type
315 */
316 /*! \def SENSOR_ALTITUDE
317 \brief GPS altitude over the ground measurement type
318 */
319 /*! \def SENSOR_HDOP
320 \brief GPS HDOP over the ground measurement type
321 */
322 /*! \def SENSOR_VDOP
323 \brief GPS VDOP over the ground measurement type
324 */
325 /*! \def SENSOR_PDOP
326 \brief GPS PDOP over the ground measurement type
327 */
328
329
330 /// State Machine
331 /*! \def SENSOR_FSM
332 \brief Finite State Machine (FSM) value
333 */
334
335
336 /// New pluviometer values
337 /*! \def SENSOR_PLV1
338 \brief pluviometer value for current hour
339 */
340 /*! \def SENSOR_PLV2
341 \brief pluviometer value for previous hour
342 */
343 /*! \def SENSOR_PLV3
344 \brief pluviometer value for last 24h
345 */
346
347 /// P&S watermarks
348 /*! \def SENSOR_SOIL_C
349 \brief watermark value for P&S connector C
350 */
351 /*! \def SENSOR_SOIL_D
352 \brief watermark value for P&S connector D
353 */
354 /*! \def SENSOR_SOIL_E
355 \brief watermark value for P&S connector E
356 */
357 /*! \def SENSOR_SOIL_F
358 \brief watermark value for P&S connector F
359 */
360
361 /// Waspnote OEM watermarks
362 /*! \def SENSOR_SOIL1
363 \brief Agriculture Board watermark1
364 */

```

```
365  /*! \def SENSOR_SOIL2
366      \brief Agriculture Board watermark2
367  */
368  /*! \def SENSOR_SOIL3
369      \brief Agriculture Board watermark3
370  */
371
372  /// DS18B20
373  /*! \def SENSOR_TCC
374      \brief DS18B20 temperature sensor
375  */
376
377  /// P&S Ultrasound depending on socket voltage ref
378  /*! \def SENSOR_US_3V3
379      \brief WRA1 Ultrasound sensor powered at 3V3
380  */
381  /*! \def SENSOR_US_5V
382      \brief WRA1 Ultrasound sensor powered at 5V
383  */
384
385
386  /// P&S Security sensors depending on socket (Security - Events board)
387  /*! \def SENSOR_LUM_D
388      \brief LDR sensor in socket D
389  */
390  /*! \def SENSOR_LUM_E
391      \brief LDR sensor in socket E
392  */
393  /*! \def SENSOR_LUM_F
394      \brief LDR sensor in socket F
395  */
396  /*! \def SENSOR_LP_D
397      \brief Liquid Presence sensor in socket D
398  */
399  /*! \def SENSOR_LP_E
400      \brief Liquid Presence sensor in socket E
401  */
402  /*! \def SENSOR_LP_F
403      \brief Liquid Presence sensor in socket F
404  */
405  /*! \def SENSOR_LL_D
406      \brief Liquid Level sensor in socketD
407  */
408  /*! \def SENSOR_LL_E
409      \brief Liquid Level sensor in socketE
410  */
411  /*! \def SENSOR_LL_F
412      \brief Liquid Level sensor in socketF
413  */
414  /*! \def SENSOR_HALL_D
415      \brief Hall Effect sensor in socket D
416  */
417  /*! \def SENSOR_HALL_E
418      \brief Hall Effect sensor in socket E
419  */
420  /*! \def SENSOR_HALL_F
421      \brief Hall Effect sensor in socket F
422  */
423
424  /// P&S liquid flow sensor depending on socket (Smart Metering)
425  /*! \def SENSOR_WF_C
426      \brief Liquid Flow sensor in socket C
427  */
428  /*! \def SENSOR_WF_E
429      \brief Liquid Flow sensor in socket E
430  */
431
432  /// Unix/Epoch timestamp
433  /*! \def SENSOR_TST
434      \brief Unix (aka Epoch) timestamp value
435  */
436
437  /// Turbidity sensor
```

```

438  /*! \def SENSOR_TURB
439      \brief Turbidity sensor
440  */
441
442
443  /// Version parameters
444  /*! \def SENSOR_VAPI
445      \brief API version
446  */
447  /*! \def SENSOR_VPROG
448      \brief Program version
449  */
450  /*! \def SENSOR_VBOOT
451      \brief Bootloader version
452  */
453
454
455  /// Gases PRO
456  /*! \def SENSOR_GP_CL2
457      \brief Chlorine measurement type
458  */
459  /*! \def SENSOR_GP_CO
460      \brief Carbon Monoxide measurement type
461  */
462  /*! \def SENSOR_GP_ETO
463      \brief Ethylene Oxide measurement type
464  */
465  /*! \def SENSOR_GP_H2
466      \brief Hydrogen measurement type
467  */
468  /*! \def SENSOR_GP_H2S
469      \brief Hydrogen Sulphide measurement type
470  */
471  /*! \def SENSOR_GP_HCL
472      \brief Hydrogen Chloride measurement type
473  */
474  /*! \def SENSOR_GP_HCN
475      \brief Hydrogen Cyanide measurement type
476  */
477  /*! \def SENSOR_GP_NH3
478      \brief Ammonia measurement type
479  */
480  /*! \def SENSOR_GP_NO
481      \brief Nitrogen Monoxide measurement type
482  */
483  /*! \def SENSOR_GP_NO2
484      \brief Nitrogen Dioxide measurement type
485  */
486  /*! \def SENSOR_GP_O2
487      \brief Oxygen measurement type
488  */
489  /*! \def SENSOR_GP_PH3
490      \brief Phosphine measurement type
491  */
492  /*! \def SENSOR_GP_SO2
493      \brief Sulfur Dioxide measurement type
494  */
495  /*! \def SENSOR_GP_CH4
496      \brief Methane and other combustible gases measurement type
497  */
498  /*! \def SENSOR_GP_O3
499      \brief Ozone measurement type
500  */
501  /*! \def SENSOR_GP_CO2
502      \brief Carbon Dioxide measurement type
503  */
504  /*! \def SENSOR_GP_TC
505      \brief Temperature Celsius measurement type
506  */
507  /*! \def SENSOR_GP_TF
508      \brief Temperature Fahrenheit measurement type
509  */
510  /*! \def SENSOR_GP_HUM

```

```

511     \brief Humidity measurement type
512 */
513 /*! \def SENSOR_GP_PRES
514     \brief Pressure measurement type
515 */
516 /*! \def SENSOR_OPC_TC
517     \brief Temperature Celsius measurement type
518 */
519 /*! \def SENSOR_OPC_TF
520     \brief Temperature Fahrenheit measurement type
521 */
522 /*! \def SENSOR_OPC_PM1
523     \brief PM1 measurement type
524 */
525 /*! \def SENSOR_OPC_PM2_5
526     \brief PM2.5 measurement type
527 */
528 /*! \def SENSOR_OPC_PM10
529     \brief PM10 measurement type
530 */
531 /*! \def SENSOR_OPC_PART
532     \brief Particle bin counter measurement type
533 */
534 /*! \def SENSOR_SWI_CA
535     \brief Calcium ion measurement type
536 */
537 /*! \def SENSOR_SWI_FL
538     \brief Fluoride ion measurement type
539 */
540 /*! \def SENSOR_SWI_BF
541     \brief Tetrafluoroborate ion measurement type
542 */
543 /*! \def SENSOR_SWI_NO
544     \brief Nitrates ion measurement type
545 */
546 /*! \def SENSOR_SWI_BR
547     \brief Bromide ion measurement type
548 */
549 /*! \def SENSOR_SWI_CL
550     \brief Chloride ion measurement type
551 */
552 /*! \def SENSOR_SWI_CU
553     \brief Cupric ion measurement type
554 */
555 /*! \def SENSOR_SWI_IO
556     \brief Iodide ion measurement type
557 */
558 /*! \def SENSOR_SWI_PB
559     \brief Lead ion measurement type
560 */
561 /*! \def SENSOR_SWI_AG
562     \brief Silver ion measurement type
563 */
564 /*! \def SENSOR_SWI_PH
565     \brief pH (for Smart Water Ions) measurement type
566 */
567 /*! \def SENSOR_TRAMA
568     \brief Sensor measurement type
569 */
570
571
572
573 // Gases
574 #define SENSOR_CO          0
575 #define SENSOR_CO2         1
576 #define SENSOR_O2          2
577 #define SENSOR_CH4         3
578 #define SENSOR_LPG         4
579 #define SENSOR_NH3         5
580 #define SENSOR_AP1         6
581 #define SENSOR_AP2         7
582 #define SENSOR_SV          8
583 #define SENSOR_NO2         9

```



```

584 #define SENSOR_O3 10
585 #define SENSOR_VOC 11
586 #define SENSOR_TCA 12
587 #define SENSOR_TFA 13
588 #define SENSOR_HUMA 14
589 #define SENSOR_PA 15
590
591 // Events
592 #define SENSOR_PW 16
593 #define SENSOR_BEND 17
594 #define SENSOR_VBR 18
595 #define SENSOR_HALL 19
596 #define SENSOR_LP 20
597 #define SENSOR_LL 21
598 #define SENSOR_LUM 22
599 #define SENSOR_PIR 23
600 #define SENSOR_ST 24
601
602 // Smart Cities
603 #define SENSOR_MCP 25
604 #define SENSOR_CDG 26
605 #define SENSOR_CPG 27
606 #define SENSOR_LD 28
607 #define SENSOR_DUST 29
608 #define SENSOR_US 30
609
610 // Smart parking
611 #define SENSOR_MF 31
612 #define SENSOR_PS 32
613
614 // Agriculture
615 #define SENSOR_TCB 33
616 #define SENSOR_TFB 34
617 #define SENSOR_HUMB 35
618 #define SENSOR_SOILT 36
619 #define SENSOR_SOIL 37
620 #define SENSOR_LW 38
621 #define SENSOR_PAR 39
622 #define SENSOR_UV 40
623 #define SENSOR_TD 41
624 #define SENSOR_SD 42
625 #define SENSOR_FD 43
626 #define SENSOR_ANE 44
627 #define SENSOR_WV 45
628 #define SENSOR_PLV 46
629
630 // Radiation
631 #define SENSOR_RAD 47
632
633 // Smart meetering
634 #define SENSOR_CU 48
635 #define SENSOR_WF 49
636 #define SENSOR_LC 50
637 #define SENSOR_DF 51
638
639 // Additional
640 #define SENSOR_BAT 52
641 #define SENSOR_GPS 53
642 #define SENSOR_RSSI 54
643 #define SENSOR_MAC 55
644 #define SENSOR_NA 56
645 #define SENSOR_NID 57
646 #define SENSOR_DATE 58
647 #define SENSOR_TIME 59
648 #define SENSOR_GMT 60
649 #define SENSOR_RAM 61
650 #define SENSOR_IN_TEMP 62
651 #define SENSOR_ACC 63
652 #define SENSOR_MILLIS 64
653
654 // Special
655 #define SENSOR_STR 65
656

```

```

657 // Meshlium
658 #define SENSOR_MBT 66
659 #define SENSOR_MWIFI 67
660
661
662 // RFID
663 #define SENSOR_UID 68
664 #define SENSOR_RB 69
665
666 // Smart Water
667 #define SENSOR_PH 70
668 #define SENSOR_ORP 71
669 #define SENSOR_DO 72
670 #define SENSOR_COND 73
671 #define SENSOR_WT 74
672 #define SENSOR_DINA 75
673 #define SENSOR_DICA 76
674 #define SENSOR_DIF 77
675 #define SENSOR_DICL 78
676 #define SENSOR_DIBR 79
677 #define SENSOR_DII 80
678 #define SENSOR_DICU2 81
679 #define SENSOR_DIK 82
680 #define SENSOR_DIMG2 83
681 #define SENSOR_DINO3 84
682
683 // Smart Libelium
684 #define SENSOR_TX_PWR 85
685 #define SENSOR_DM_ST 86
686 #define SENSOR_DM_SP 87
687 #define SENSOR_LUX 88
688
689 // GPS
690 #define SENSOR_SPEED 89
691 #define SENSOR_COURSE 90
692 #define SENSOR_ALTITUDE 91
693 #define SENSOR_HDOP 92
694 #define SENSOR_VDOP 93
695 #define SENSOR_PDOP 94
696
697 // Finite State Machine status
698 #define SENSOR_FSM 95
699
700 // New pluviometer values
701 #define SENSOR_PLV1 96
702 #define SENSOR_PLV2 97
703 #define SENSOR_PLV3 98
704
705 // P&S watermark sensors (Smart Agriculture)
706 #define SENSOR_SOIL_C 99
707 #define SENSOR_SOIL_D 100
708 #define SENSOR_SOIL_E 101
709 #define SENSOR_SOIL_F 102
710
711 // Waspmote OEM watermark sensors
712 #define SENSOR_SOIL1 103
713 #define SENSOR_SOIL2 104
714 #define SENSOR_SOIL3 105
715
716 // DS18B20
717 #define SENSOR_TCC 106
718
719 // P&S Ultrasound depending on socket voltage ref (Smart Cities & Smart Metering)
720 #define SENSOR_US_3V3 107
721 #define SENSOR_US_5V 108
722
723 // P&S Security sensors depending on socket (Smart Security)
724 #define SENSOR_LUM_D 109
725 #define SENSOR_LUM_E 110
726 #define SENSOR_LUM_F 111
727 #define SENSOR_LP_D 112
728 #define SENSOR_LP_E 113
729 #define SENSOR_LP_F 114

```

```

730 #define SENSOR_LL_D      115
731 #define SENSOR_LL_E      116
732 #define SENSOR_LL_F      117
733 #define SENSOR_HALL_D    118
734 #define SENSOR_HALL_E    119
735 #define SENSOR_HALL_F    120
736
737 // P&S liquid flow sensor depending on socket (Smart Metering)
738 #define SENSOR_WF_C      121
739 #define SENSOR_WF_E      122
740
741 // Unix/Epoch timestamp
742 #define SENSOR_TST      123
743
744 // Turbidity sensor
745 #define SENSOR_TURB      124
746
747 // Version parameters
748 #define SENSOR_VAPI      125
749 #define SENSOR_VPROG      126
750 #define SENSOR_VBOOT      127
751
752
753 // Gases PRO
754 #define SENSOR_GP_CL2    128
755 #define SENSOR_GP_CO     129
756 #define SENSOR_GP_ETO    130
757 #define SENSOR_GP_H2     131
758 #define SENSOR_GP_H2S    132
759 #define SENSOR_GP_HCL    133
760 #define SENSOR_GP_HCN    134
761 #define SENSOR_GP_NH3    135
762 #define SENSOR_GP_NO     136
763 #define SENSOR_GP_NO2    137
764 #define SENSOR_GP_O2     138
765 #define SENSOR_GP_PH3    139
766 #define SENSOR_GP_SO2    140
767 #define SENSOR_GP_CH4    141
768 #define SENSOR_GP_O3     142
769 #define SENSOR_GP_CO2    143
770 #define SENSOR_GP_TC     144
771 #define SENSOR_GP_TF     145
772 #define SENSOR_GP_HUM    146
773 #define SENSOR_GP_PRES    147
774
775 // OPCN2 Dust Sensor
776 #define SENSOR_OPC_TC      148
777 #define SENSOR_OPC_TF      149
778 #define SENSOR_OPC_P       150
779 #define SENSOR_OPC_PM1     151
780 #define SENSOR_OPC_PM2_5   152
781 #define SENSOR_OPC_PM10    153
782 #define SENSOR_OPC_PART    154
783
784 // Smart Water Ions
785 #define SENSOR_SWI_CA      155
786 #define SENSOR_SWI_FL      156
787 #define SENSOR_SWI_FB      157
788 #define SENSOR_SWI_NO      158
789 #define SENSOR_SWI_BR      159
790 #define SENSOR_SWI_CL      160
791 #define SENSOR_SWI_CU      161
792 #define SENSOR_SWI_IO      162
793 #define SENSOR_SWI_PB      163
794 #define SENSOR_SWI_AG      164
795 #define SENSOR_SWI_PH      165
796
797 // P&S Smart Water sensors depending on socket (Smart Water)
798 #define SENSOR_PH_A        166
799 #define SENSOR_PH_B        167
800 #define SENSOR_PH_C        168
801 #define SENSOR_ORP_A       169
802 #define SENSOR_ORP_B       170

```

```

803 #define SENSOR_ORP_C      171
804 #define SENSOR_TRAMA      200
805
806 // define MACROS in order to manage bits inside Bytes
807 #define bitRead(value, bit) (((value) >> (bit)) & 0x01)
808 #define bitSet(value, bit) ((value) |= (1UL << (bit)))
809 #define bitClear(value, bit) ((value) &= ~(1UL << (bit)))
810 #define bitWrite(value, bit, bitvalue) (bitvalue ? bitSet(value, bit) :
    bitClear(value, bit))
811
812
813 /*! \def MAX_FRAME
814     \brief maximum size in bytes of the frame
815 */
816 #define MAX_FRAME 150
817
818
819
820 /*! \def UNICAST_16B
821     \brief
822 */
823 /*! \def UNICAST_64B
824     \brief
825 */
826 /*! \def BROADCAST_MODE
827     \brief
828 */
829 #define UNICAST_16B      0
830 #define UNICAST_64B      1
831 #define BROADCAST_MODE  2
832
833
834
835 /*! \def EXAMPLE_FRAME
836     \brief
837 */
838 /*! \def TIMEOUT_FRAME
839     \brief
840 */
841 /*! \def EVENT_FRAME
842     \brief
843 */
844 /*! \def ALARM_FRAME
845     \brief
846 */
847 /*! \def SERVICE1_FRAME
848     \brief
849 */
850 /*! \def SERVICE2_FRAME
851     \brief
852 */
853 #define EXAMPLE_FRAME    0
854 #define TIMEOUT_FRAME    1
855 #define EVENT_FRAME      2
856 #define ALARM_FRAME      3
857 #define SERVICE1_FRAME   4
858 #define SERVICE2_FRAME   5
859 #define SET_TIME_FRAME   6
860
861 /*! \def AES128_ECB_FRAME
862     \brief Encrypted frame using AES-128 key size and ECB mode
863 */
864 /*! \def AES192_ECB_FRAME
865     \brief Encrypted frame using AES-192 key size and ECB mode
866 */
867 /*! \def AES256_ECB_FRAME
868     \brief Encrypted frame using AES-256 key size and ECB mode
869 */
870 #define AES128_ECB_FRAME  97
871 #define AES192_ECB_FRAME  98
872 #define AES256_ECB_FRAME  99
873
874

```

```

875
876  /*! \def TYPE_UINT8
877      \brief TYPE_UINT8 defines the constant for uint8_t variables types (1 Byte)
878  */
879  /*! \def TYPE_INT
880      \brief TYPE_INT defines the constant for int (int16_t) variables types (2 Bytes)
881  */
882  /*! \def TYPE_FLOAT
883      \brief TYPE_FLOAT defines the constant for double/float variables types (4 Bytes)
884  */
885  /*! \def TYPE_CHAR
886      \brief TYPE_CHAR defines the constant for char* (strings) variables types
887          (variable Bytes)
888  */
889  /*! \def TYPE_ULONG
890      \brief TYPE_ULONG defines the constant for unsigned long int variables types (4
891          Bytes)
892  */
891  #define TYPE_UINT8          0
892  #define TYPE_INT            1
893  #define TYPE_FLOAT          2
894  #define TYPE_CHAR           3
895  #define TYPE_ULONG          4
896
897
898
899  /*! \def ASCII
900      \brief ASCII frame mode
901  */
902  /*! \def BINARY
903      \brief BINARY frame mode
904  */
905  #define BINARY              0
906  #define ASCII                1
907  #define ENCRYPTED_FRAME     2
908
909
910  //! Variable : Wasmote serial id
911  /*!
912  */
913  extern volatile unsigned long _serial_id;
914
915
916  /*****
917   * Class
918   *****/
919
920  //! WaspFrame Class
921  /*!
922      WaspFrame Class defines all the variables and functions used to create the
923      sensor frames in Wasmote
924  */
925  class WaspFrame
926  {
927
928  private:
929
930      //! Function : store the sequence number in EEPROM
931      /*! This function stores the sequence number in the EEPROM address specified
932          * by the SEQUENCE_ADDR address.
933          */
934      void storeSequence(uint8_t seqNumber);
935
936      //! Function : read the sequence number from EEPROM
937      /*! This function reads the sequence number from the EEPROM address specified
938          * by the SEQUENCE_ADDR address.
939          */
940      uint8_t readSequence(void);
941
942      //! Function : check if maximum length is reached
943      /*! Check if maximum length is reached when introducing a new frame field
944          */
945      uint8_t checkLength(int sum);

```

```

946
947     //!< Variable : sequence number
948     /*!
949     This is the frame sequence number in order to control if any packet is lost
950     */
951     uint8_t sequence;
952
953     //!< Variable : number of sensor values within the frame
954     /*!
955     */
956     uint8_t numFields;
957
958     //!< Variable : frame format: ASCII (0) or BINARY (1)
959     /*!
960     */
961     uint8_t _mode;
962
963     //!< Variable : maximum frame size
964     /*!
965     */
966     uint16_t _maxSize;
967
968     //!< Variable : buffer for Wasp mote ID. 16B maximum
969     /*!
970     */
971     char _waspMoteID[17];
972
973 public:
974
975     //!< class constructor
976     /*!
977     Initialize the class attributes
978     \param void
979     \return void
980     */
981     WaspFrame();
982
983     //!< Function : set the frame maximum size
984     /*! This function sets the frame maximum size
985     \param uint8_t size size to set as maximum size. It can't exceed the
986           MAX_FRAME constant
987     \return void
988     */
989     void setFrameSize( uint8_t size);
990
991     //!< Function : set the frame maximum size
992     /*! This function sets the frame maximum size depending on the protocol,
993     addressing, linkEncryption mode, AESEncryption.
994     \param uint8_t protocol defines the protocol used
995     \param uint8_t linkEncryption defines if XBee encryption is enabled or not
996     \param uint8_t AESEncryption defines if AES encryption is used or not
997     \return void
998     */
999     void setFrameSize( uint8_t protocol,
1000                       uint8_t linkEncryption,
1001                       uint8_t AESEncryption);
1002
1003     //!< Function : set the frame maximum size
1004     /*! This function sets the frame maximum size depending on the protocol,
1005     addressing, linkEncryption mode, AESEncryption.
1006     \param uint8_t protocol defines the protocol used
1007     \param uint8_t addressing defines the addressing used
1008     \param uint8_t linkEncryption defines if XBee encryption is enabled or not
1009     \param uint8_t AESEncryption defines if AES encryption is used or not
1010     \return void
1011     */
1012     void setFrameSize( uint8_t protocol,
1013                       uint8_t addressing,
1014                       uint8_t linkEncryption,
1015                       uint8_t AESEncryption);
1016
1017     //!< Function : get the frame maximum size
1018     /*! This function gets the frame maximum size

```

```

1019 \return uint8_t indicating the value of _maxSize
1020 */
1021 uint8_t getFrameSize( void );
1022
1023 //!< Function : creates a new frame
1024 /*! This function creates a new ASCII frame getting the mote ID from the
1025 * EEPROM memory.
1026 */
1027 void createFrame(uint8_t mode);
1028
1029 //!< Function : creates a new frame
1030 /*! This function creates a new frame.
1031 \param uint8_t mode specifies the frame mode: BINARY or ASCII
1032 \return char* moteID defines the mote Identifier
1033 */
1034 void createFrame(uint8_t mode, char* moteID);
1035
1036 //!< Function : creates a encrypted frame
1037 /*! This function creates a new frame, encrypting the actual contents of the
1038 * Wasmote Frame with the AES-key specified as input. The encrypted message
1039 * becomes the payload of the new encapsulated frame
1040 \param uint16_t AESmode: specifies the AES key mode: 128, 192 or 256
1041 \param char* password: specifies the AES key as a string
1042 \return '1' if OK; '0' otherwise
1043 */
1044 uint8_t encryptFrame( uint16_t AESmode, char* password );
1045
1046
1047 //!< Function : set the frame type
1048 /*! This function sets the frame type (fourth byte of the frame header)
1049 \param uint8_t type specifies the frame type:
1050     EXAMPLE_FRAME
1051     TIMEOUT_FRAME
1052     EVENT_FRAME
1053     ALARM_FRAME
1054     SERVICE1_FRAME
1055     SERVICE2_FRAME
1056 \return void
1057 */
1058 void setFrameType(uint8_t type);
1059
1060 //!< Function : shows the frame
1061 /*! This function prints the actual frame
1062 */
1063 void showFrame(void);
1064
1065 int8_t addSensor(uint8_t type, int value);
1066 int8_t addSensor(uint8_t type, unsigned long value);
1067 int8_t addSensor(uint8_t type, double value);
1068 int8_t addSensor(uint8_t type, double value, int N);
1069 int8_t addSensor(uint8_t type, char* str);
1070 /* afegit vol dir que son els valors adaptats per poder enviar les comandes
agrupades així que s'han creat de diferents per tal
1071 de fer diferents enviaments aquests s'adapten als valors dels waspmote fent
referència al WaspFrame.cpp que s'ha creat*/
1072 int8_t addSensor(uint8_t type, double val1,double val2, double val3,double
val4,int val5); // afegit
1073 int8_t addSensor(uint8_t type, double val1,double val2, double val3,int val4);
// afegit
1074 int8_t addSensor(uint8_t type, double val1,double val2, double val3,double
val4,double val5,int val6); // afegit
1075 int8_t addSensor(uint8_t type, double val1,double val2, double val3,int val4,int
val5,int val6,int val7); // afegit
1076 int8_t addSensor(uint8_t type, double val1,double val2, int val3,int val4,int
val5,int val6,int val7,int val8);
1077 int8_t addSensor(uint8_t type, double val1, double val2);
1078 int8_t addSensor(uint8_t type, unsigned long val1, unsigned long val2);
1079 int8_t addSensor(uint8_t type, uint8_t val1, uint8_t val2, uint8_t val3);
1080 int8_t addSensor(uint8_t type, uint8_t val1, uint8_t val2, uint8_t val3, int
val4);
1081 int8_t addSensor(uint8_t type, int val1,int val2,int val3);
1082 int8_t addSensor(uint8_t type, double val1,double val2,double val3);
1083

```

```
1084     int8_t addSensor(uint8_t type, double val1,double val2, double val3,double
1085     val4,double val5,double val6,double val7,double val8);// afegit
1086
1087     int8_t checkFields(uint8_t type, uint8_t typeVal, uint8_t fields);
1088
1089     void setID(char* moteID);
1090     void getID(char* moteID);
1091
1092     void decrementSequence(void);
1093
1094     /*! Variable : buffer where the frame is created in
1095     */
1096     uint8_t buffer[MAX_FRAME+1];
1097
1098     /*! Variable : length of the frame
1099     */
1100     uint16_t length;
1101
1102     int8_t addTimestamp(void);
1103 };
1104
1105 extern WaspFrame frame;
1106
1107 #endif
1108
1109
1110
```