

5+

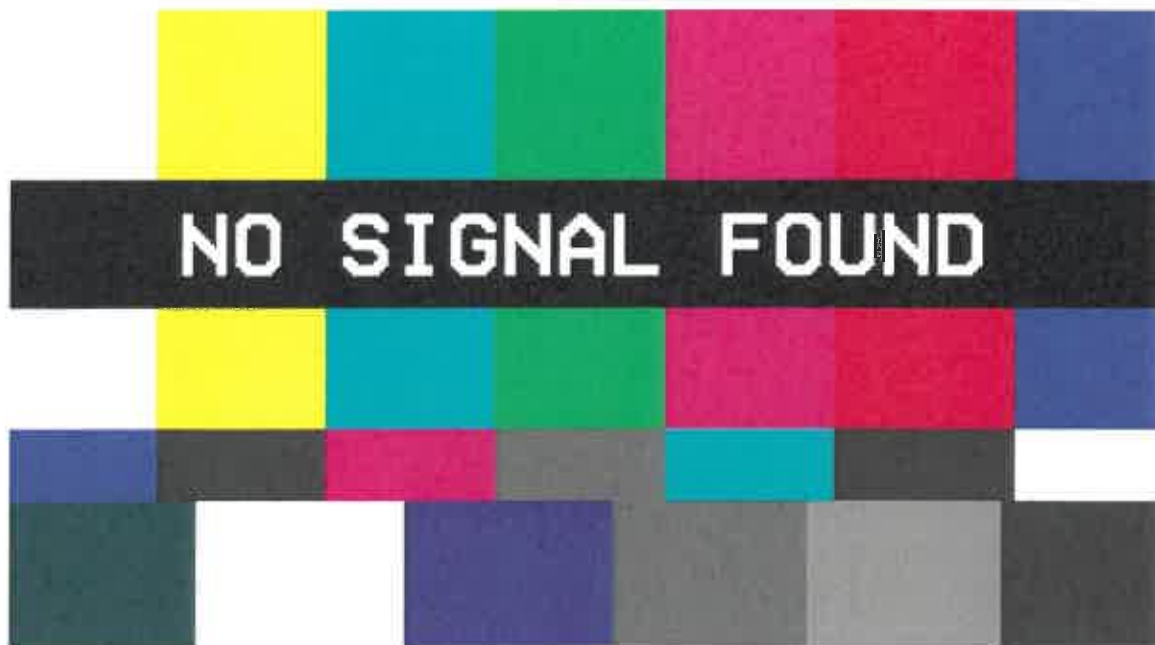


Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Systeme embarqués

Auteurs :
Marc ROTEN
Sven ROUVINEZ

Professeur :
Daniel GACHET



16 décembre 2018



Table des matières

1	Heure de travail	2
2	Introduction	2
3	Synthèse	2
4	Quelle est la signification du qualificatif volatile et quelle est son utilité quand il est associé à un pointeur ?	2
5	Comment sont placés les champs (membres) d'une structure dans la mémoire ?	3
6	Comment peut-on efficacement définir les registres d'un contrôleur de périphérique situés dans l'espace d'adressage du μP ainsi que leur contenu en C ?	3
7	Comment peut-on accéder ces registres ?	3
8	Comment générer des nombres aléatoires ?	4
9	A la fréquence maximale (24MHz), le compteur du timer ne permet de compter le temps que sur un intervalle de 3 minutes environ. Décrivez l'algorithme à mettre en place si l'on souhaite compter sur plusieurs années avec la même granularité	4
10	Conclusion	5

1 Heure de travail

12 heures ✓ *est le même ?*

2 Introduction

Dans ce travail-ci, nous devrons utiliser un nouveau composant, le DMTIMER, et d'autres composants tels que les boutons et l'écran LED. Le but avec ce DMTIMER est de faire un minijeu dont le but sera de tester nos reflexes. On devra presser le bouton S2, et dès que le signal se relâche le bouton S2 apparaît, on relâche le dit-bouton et la différence de temps entre les deux sera le temps que l'on affichera sur le LCD.

3 Synthèse

Sven *Acquis*

- Déclaration de struct pour travailler avec un timer
- Initialisation et écriture sur l'écran OLED

Marc *Acquis*

- Découpage et réflexion d'un problème grâce à une machine d'état déterministe (FSM).
- Lecture de documentation et compréhension du fonctionnement à partir de cette dite documentation.
- Compréhension du fonctionnement de DMTIMER des boutons et du display LCD.
- utilisation de la fonction Debug d'Eclipse sur notre BeagleBone.

4 Quelle est la signification du qualificatif volatile et quelle est son utilité quand il est associé à un pointeur ?

Permet d'indiquer au compilateur de ne pas faire d'optimisation ✓

L'utilité de lier avec un pointeur et de ne pas dépendre de l'exécution du code pour que la valeur change, dans le cas de notre compteur, par exemple, même si le processus travaille sur autre chose, notre timer va continuer à s'incrémenter ✓

5 Comment sont placés les champs (membres) d'une structure dans la mémoire ?

De façon continue selon la taille du type choisit ✓

6 Comment peut-on efficacement définir les registres d'un contrôleur de périphérique situés dans l'espace d'adressage du µP ainsi que leur contenu en C ?

En utilisant des struct

```
struct timer_reg {  
    uint32_t tidr;    //00  
    uint32_t res1[3]; //04--0f  
    uint32_t tiocp_cfg; //10  
    uint32_t res2[3]; //14--1f  
    uint32_t irq_eoi;  
    uint32_t irqstatus_raw;  
    uint32_t irqstatus;  
    uint32_t irqenable_set;  
    uint32_t irqenable_clr;  
    uint32_t irqwakeen;  
    uint32_t tclr;  
    uint32_t tcerr;  
    uint32_t tldr;  
    uint32_t ttgr;  
    uint32_t twps;  
    uint32_t tmar;  
    uint32_t tcar1;  
    uint32_t tsicr;  
    uint32_t tcar2;  
};
```

7 Comment peut-on accéder ces registres ?

En affectant une variable *→ en pointeur* static volatile

```
static volatile struct timer_reg* dmtimer[] = {  
    (volatile struct timer_reg*) 0x48040000,
```

```
        (volatile struct timer_reg*) 0x48042000 ,  
        (volatile struct timer_reg*) 0x48044000 ,  
        (volatile struct timer_reg*) 0x48046000 ,  
        (volatile struct timer_reg*) 0x48048000 ,  
        (volatile struct timer_reg*) 0x4804A000  
};  
  
uint32_t dmtimer1_get_counter(enum dmtimer_timers timer){  
    volatile struct timer_reg* ctrl = dmtimer[timer];  
    return ctrl->tcr;  
}
```

8 Comment générer des nombres aléatoires ?

Cette fonction permet de générer des nombres aléatoires selon un intervalle donné

```
rand();  
uint32_t random_time() {  
    return random() % ((MAX_VALUE_TIMER + MIN_VALUE_TIMER) + 1)  
        + MAX_VALUE_TIMER;  
}
```

9 A la fréquence maximale (24MHz), le compteur du timer ne permet de compter le temps que sur un intervalle de 3 minutes environ. Décrivez l'algorithme à mettre en place si l'on souhaite compter sur plusieurs années avec la même granularité

Pour calculer le temps écoulé, on calcule le delta entre le moment où le timer a été démarré et le moment où l'on veut qu'il s'arrête.

Dans notre cas, la fréquence étant de 24Mhz, un timer peut compter jusqu'à environ 3 minutes donc pour aller plus loin il faut faire un pointage maximum jusqu'à 3 minutes afin d'éviter de louper une interruption, et donc d'incrémenter une variable qui garde le nombre de fois que l'on a contrôlé notre timer. Pour calculer le delta, il faut utiliser :

$$\Delta = sp - st$$

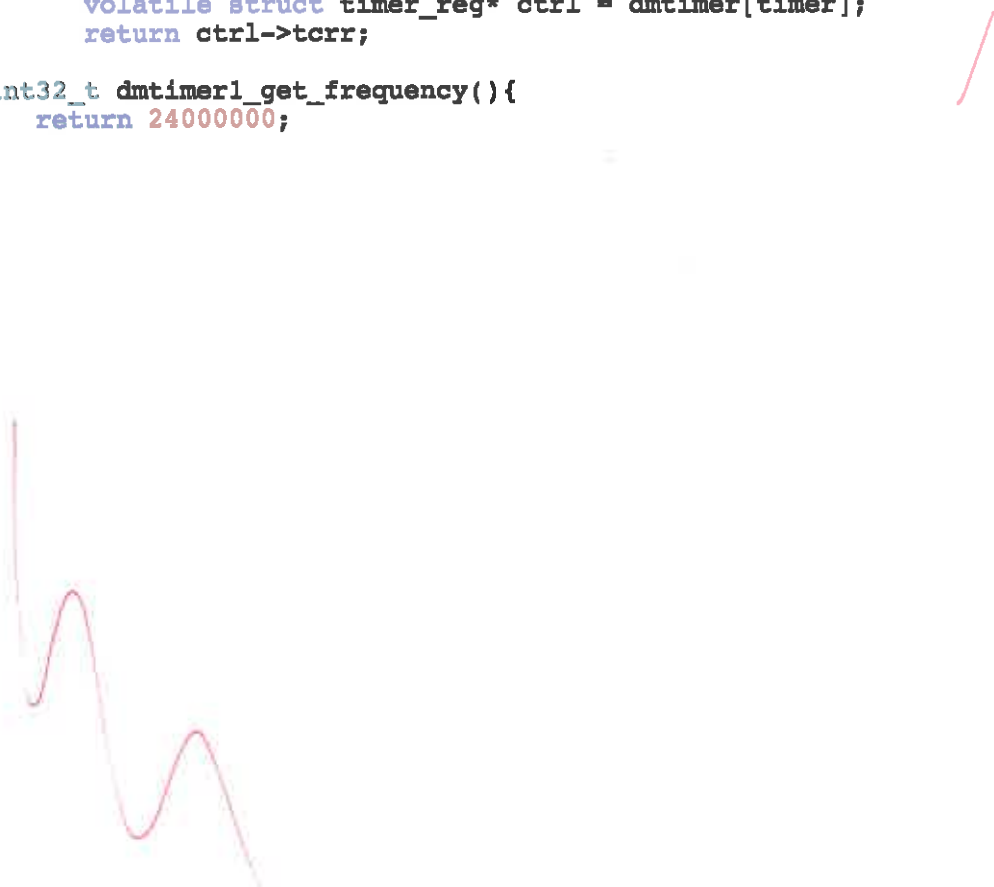
, où *sp* est le stop time et *st* le start time

10 Conclusion

On a pu au travers de ce TP, concevoir les blocs DMTIMER et rajouter les fonctions displayString et displayChar dans notre fichier display.c. Ce TP s'est bien passé et était bien documenté. ✓

```
1 file: 5e-groupe/tp.04/timer.c
2 /**
3  * Copyright 2018 University of Applied Sciences Western Switzerland / Fribourg
4  *
5  * Licensed under the Apache License, Version 2.0 (the "License");
6  * you may not use this file except in compliance with the License.
7  * You may obtain a copy of the License at
8  *
9  *     http://www.apache.org/licenses/LICENSE-2.0
10 *
11 * Unless required by applicable law or agreed to in writing, software
12 * distributed under the License is distributed on an "AS IS" BASIS,
13 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14 * See the License for the specific language governing permissions and
15 * limitations under the License.
16 *
17 * Project:      HEIA-FR / Embedded Systems 4 Laboratory
18 *
19 *
20 * Author:       Marc Roten / Sven Rouvinez
21 * Date:         Dec 3, 2018
22 */
23
24 #include <stdio.h>
25 #include <stdint.h>
26 #include <stdbool.h>
27 #include <am335x_clock.h>
28 #include "timer.h"
29
30
31 // DMTimer TIOCP_CFG register bit definition
32 #define TIOCP_CFG_SOFTRESET      (1<<0)
33
34 // DMTimer TISTAT register bit definition
35 #define TISTAT_RESETDONE        (1<<0)
36
37 // DMTimer TCLR register bit definition
38 #define TCLR_ST                  (1<<0)
39 #define TCLR_AR                  (1<<1)
40
41
42
43
44
45 struct timer_reg {
46     uint32_t tidr;      //00
47     uint32_t res1[3];   //04--0f
48     uint32_t tiocp_cfg; //10
49     uint32_t res2[3];   //14--1f
50     uint32_t irq_eoi;
51     uint32_t irqstatus_raw;
52     uint32_t irqstatus;
53     uint32_t irgenable_set;
54     uint32_t irgenable_clr;
55     uint32_t irqwakeen;
56     uint32_t tclr;
57     uint32_t tcrr;
58     uint32_t tldr;
59     uint32_t ttgr;
60     uint32_t twps;
61     uint32_t tmar;
62     uint32_t tcar1;
63     uint32_t tsicr;
64     uint32_t tcar2;
65 };
66
67 static volatile struct timer_reg* dmtimer[] = {
68     (volatile struct timer_reg*) 0x48040000,
69     (volatile struct timer_reg*) 0x48042000,
```

```
70     (volatile struct timer_reg*) 0x48044000,  
71     (volatile struct timer_reg*) 0x48046000,  
72     (volatile struct timer_reg*) 0x48048000,  
73     (volatile struct timer_reg*) 0x4804A000  
74 };  
75  
76 static const enum am335x_clock_timer_modules timer2clock[] = {  
77     AM335X_CLOCK_TIMER2, AM335X_CLOCK_TIMER3, AM335X_CLOCK_TIMER4,  
78     AM335X_CLOCK_TIMER5, AM335X_CLOCK_TIMER6, AM335X_CLOCK_TIMER7  
79 };  
80  
81  
82 void dmtimer1_init(enum dmtimer_timers timer){  
83     am335x_clock_enable_timer_module(timer2clock[timer]);  
84  
85     volatile struct timer_reg* ctrl = dmtimer[timer];  
86     ctrl->tiocp_cfg = TIOCP_CFG_SOFTRESET;  
87     while ((ctrl->tiocp_cfg & TIOCP_CFG_SOFTRESET) != 0)  
88         ;  
89  
90     ctrl->tldr = 0;  
91     ctrl->tcrr = 0;  
92     ctrl->ttgr = 0;  
93     ctrl->tclr = TCLR_AR | TCLR_ST;  
94 }  
95  
96  
97  
98 uint32_t dmtimer1_get_counter(enum dmtimer_timers timer){  
99     volatile struct timer_reg* ctrl = dmtimer[timer];  
100     return ctrl->tcrr;  
101 }  
102 uint32_t dmtimer1_get_frequency(){  
103     return 24000000;  
104 }  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138
```



139
140
141
142



```
143 file: 5e-groupe/tp.04/buttons.c
144 /**
145  * Copyright 2018 University of Applied Sciences Western Switzerland / Fribourg
146  *
147  * Licensed under the Apache License, Version 2.0 (the "License");
148  * you may not use this file except in compliance with the License.
149  * You may obtain a copy of the License at
150  *
151  * http://www.apache.org/licenses/LICENSE-2.0
152  *
153  * Unless required by applicable law or agreed to in writing, software
154  * distributed under the License is distributed on an "AS IS" BASIS,
155  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
156  * See the License for the specific language governing permissions and
157  * limitations under the License.
158  *
159  * Project:      HEIA-FR / Embedded Systems
160  *
161  *
162  * Author:       Marc Roten / Sven Rouvinez
163  * Date:         3 Dec 2018
164  */
165
166
167
168 #include "buttons.h"
169 #include <am335x_gpio.h>
170
171 // pin definition for buttons access
172 #define BTN_GPIO    AM335X_GPIO1
173 #define S1_PIN (15)
174 #define S2_PIN (16)
175 #define S3_PIN (17)
176
177 void button_init() {
178     //init of GPIO
179     am335x_gpio_init(BTN_GPIO);
180     //init of the buttons as Input
181     am335x_gpio_setup_pin_in(BTN_GPIO, S1_PIN, AM335X_GPIO_PULL_NONE, 0);
182     am335x_gpio_change_state(BTN_GPIO, S1_PIN, 1);
183     am335x_gpio_setup_pin_in(BTN_GPIO, S2_PIN, AM335X_GPIO_PULL_NONE, 0);
184     am335x_gpio_change_state(BTN_GPIO, S2_PIN, 1);
185     am335x_gpio_setup_pin_in(BTN_GPIO, S3_PIN, AM335X_GPIO_PULL_NONE, 0);
186     am335x_gpio_change_state(BTN_GPIO, S2_PIN, 1);
187 }
188
189
190 bool button_s1_is_pressed() {
191     return !am335x_gpio_get_state(BTN_GPIO, S1_PIN);
192 }
193
194 bool button_s2_is_pressed() {
195     return !am335x_gpio_get_state(BTN_GPIO, S2_PIN);
196 }
197 bool button_s3_is_pressed() {
198     return !am335x_gpio_get_state(BTN_GPIO, S3_PIN);
199 }
200
201
202
```

initial pour des inputs

```
203 file: 5e-groupe/tp.04/timer.h
204 #pragma once
205 /**
206  * Copyright 2018 University of Applied Sciences Western Switzerland / Fribourg
207  *
208  * Licensed under the Apache License, Version 2.0 (the "License");
209  * you may not use this file except in compliance with the License.
210  * You may obtain a copy of the License at
211  *
212  * http://www.apache.org/licenses/LICENSE-2.0
213  *
214  * Unless required by applicable law or agreed to in writing, software
215  * distributed under the License is distributed on an "AS IS" BASIS,
216  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
217  * See the License for the specific language governing permissions and
218  * limitations under the License.
219  *
220  * Author:      Marc Roten / Sven Rouvinez
221  * Date:        Dec 3, 2018
222  */
223 #ifndef TIMER_H_
224 #define TIMER_H_
225
226 #include <stdint.h>
227
228
229 enum dmtimer_timers{
230     DMTIMER2,
231     DMTIMER3,
232     DMTIMER4,
233     DMTIMER5,
234     DMTIMER6,
235     DMTIMER7
236 };
237
238
239 /**
240  * we init our timer to zero
241  */
242 extern void dmtimer1_init(enum dmtimer_timers timer);
243
244
245 /**
246  * we get the value of our counter
247  */
248 extern uint32_t dmtimer1_get_counter(enum dmtimer_timers timer);
249
250
251 /**
252  * get the frequency
253  */
254 extern uint32_t dmtimer1_get_frequency();
255
256
257
258 #endif
259
260
```

```
261 file: 5e-groupe/tp.04/main.c
262 /**
263  * Copyright 2018 University of Applied Sciences Western Switzerland / Fribourg
264  *
265  * Licensed under the Apache License, Version 2.0 (the "License");
266  * you may not use this file except in compliance with the License.
267  * You may obtain a copy of the License at
268  *
269  * http://www.apache.org/licenses/LICENSE-2.0
270  *
271  * Unless required by applicable law or agreed to in writing, software
272  * distributed under the License is distributed on an "AS IS" BASIS,
273  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
274  * See the License for the specific language governing permissions and
275  * limitations under the License.
276  */
277 <<<<<< HEAD
278 * Project:      HEIA-FR / Embedded Systems
279 *
280 *
281 * Author:       Marc Roten / Sven Rouvinez
282 * Date:         3 Dec 2018
283 */
284
285 #include <stdio.h>
286 #include <stdlib.h>
287 #include <stdint.h>
288 #include <stdbool.h>
289
290 #include "timer.h"
291 #include "buttons.h"
292 #include "display.h"
293 #include "oled.h"
294
295 #define MAX_VALUE_TIMER 60000000 //2500*24'000'000*10-3 = 60000000
296 #define MIN_VALUE_TIMER 12000000 //500 = 60000000/5 = 12000000
297
298
299
300 #define BLACK 0
301 #define RED 0xff000
302 #define WHITE 0xfffff
303 #define GREEN 0x00cc0
304 #define ORANGE 0xFC000
305 #define UNIT_TIME 1000
306
307 #define S1 (15)
308 #define S2 (16)
309 #define S3 (17)
310
311 enum states {
312     IDLE, PRESSED, ERROR, RELEASED, SCORE,
313 };
314
315 uint32_t random_time() {
316     rand();
317     return random() % (MAX_VALUE_TIMER + 1 - MIN_VALUE_TIMER) + MIN_VALUE_TIMER;
318 }
319
320
321 void delay(int value){
322     while (value > 0) value--;
323 }
324
325
326 void init() {
327     button_init();
328     display_init();
329 }
```

*une bonne idée
car la fréquence !
- you wait change !*

```
330 }
331
332 int main() {
333     init();
334     uint32_t rand = 0;
335
336     enum states state = IDLE;
337     uint32_t resTime = 0;
338     uint32_t timerReact = 0;
339     double displayTime = 0;
340
341     while (1) {
342         switch (state) {
343             case IDLE:
344
345                 display_string(0, 72, "ready", WHITE, BLACK);
346
347                 rand = random_time();
348                 if (button_s2_is_pressed()) { //cas ou on presse le bouton 2
349                     display_clear();
350                     display_string(0, 60, "GET READY", ORANGE, BLACK);
351                     state = PRESSED;
352                     dmtimer1_init(DMTIMER2);
353                     dmtimer1_init(DMTIMER3);
354                 }
355                 break;
356             case PRESSED:
357
358                 if (button_s2_is_pressed()) {
359                     display_string(0, 60, "GET READY", ORANGE, BLACK);
360                     while ((timerReact = dmtimer1_get_counter(DMTIMER2)) < rand) {
361                         if (!button_s2_is_pressed()) {
362                             state = ERROR;
363                             break;
364                         }
365                     }
366                     state = RELEASED;
367                     break;
368                 }
369                 else{
370                     state = ERROR;
371                     break;
372                 }
373             case ERROR:
374                 while(!button_s2_is_pressed()) {
375                     display_string(24, 72, "ERROR", RED, BLACK);
376                 }
377                 state = IDLE;
378                 display_clear();
379                 break;
380             case RELEASED: //affiche le moment ou le joueur se retire
381                 while (button_s2_is_pressed()) {
382                     display_string(0, 60, "RELEASE NOW", GREEN, BLACK);
383                     resTime = dmtimer1_get_counter(DMTIMER3) - timerReact;
384                 }
385                 state = SCORE;
386                 break;
387             case SCORE:
388                 printf("\n"); // do not remove = bug in C https://stackoverflow.com/question
389                 char myResString[72];
390
391                 displayTime = ((double) resTime / dmtimer1_get_frequency())*UNIT_TIME;
```

faites des petits forêts pour l'init.
ou vous le codez pour l'init.

```
399
400     sprintf(myResString, "%d ms", (int)displayTime ); //(passage secondes à mili
401
402
403     while(!button_s2_is_pressed()){
404         display_string(0, 24, "react time", RED, BLACK);
405         display_string(0, 16, myResString, RED, BLACK);
406     }
407     display_clear();
408     state = IDLE;
409
410     break;
411 default:
412     state = IDLE;
413     break;
414 }
415 }
416
417 return 0;
418 }
419 ;
420
```

```
421 file: 5e-groupe/tp.04/buttons.h
422 /**
423  * Copyright 2018 University of Applied Sciences Western Switzerland / Fribourg
424  *
425  * Licensed under the Apache License, Version 2.0 (the "License");
426  * you may not use this file except in compliance with the License.
427  * You may obtain a copy of the License at
428  *
429  *     http://www.apache.org/licenses/LICENSE-2.0
430  *
431  * Unless required by applicable law or agreed to in writing, software
432  * distributed under the License is distributed on an "AS IS" BASIS,
433  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
434  * See the License for the specific language governing permissions and
435  * limitations under the License.
436  *
437  * Project:      HEIA-FR / Embedded Systems
438  *
439  *
440  * Author:       Marc Roten / Sven Rouvinez
441  * Date:         3 Dec 2018
442  */
443 #pragma once
444 #ifndef BUTTONS_H_
445 #define BUTTONS_H_
446
447 #include <stdbool.h>
448
449 /**
450  * method to initialize the resources of the buttons
451  * this method shall be called prior any other.
452  */
453 extern void button_init();
454
455
456 /**
457  * method to know if the button S1 is pressed
458  */
459 extern bool button_s1_is_pressed();
460
461
462 /**
463  * method to know if the button S2 is pressed
464  */
465 extern bool button_s2_is_pressed();
466
467
468 /**
469  * method to know if the button S3 is pressed
470  */
471 extern bool button_s3_is_pressed();
472
473
474 #endif /* BUTTON_H */
475
476
477
478
479
480
481
```

