LedMonostable

Generated by Doxygen 1.9.1

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 2

File Documentation

2.1 inc/Aplication.h File Reference

: Functions used in main

```
#include "LPC845.h"
#include "GPIO_FW.h"
#include "GPIO_SW.h"
#include "SwitchMatrix_FW.h"
#include "SYSCON_FW.h"
#include "SysTick_FW.h"
```

Macros

- #define **RED_LIGTH** PORT0,13
- #define **YELLOW_LIGTH** PORT0,14
- #define **GREEN_LIGTH** PORT0,15
- #define **RESET** 0x00
- #define R_STAGE 0x01

  Red on.
- #define RY_STAGE 0x02

  Red,Yellow on.
- #define G_STAGE 0x03

  Green on.
- #define Y_STAGE 0x04

  Yellow on.
- #define R_TICK 25∗TICK_OUT_1S

  Red 25s.
- #define RY_TICK 2∗TICK_OUT_1S

  Red,Yellow 2s.
- #define G_TICK 50∗TICK_OUT_1S

  Green 50s.
- #define Y_TICK 5∗TICK_OUT_1S

  Yellow 5s.

Functions

- void LPC_Init (void)
  : Initialize the board
- void GPIO_Init (void)
  : Initialize the GPIO
- void Semaphore (void)
  : Control of the ligths

2.1.1 Detailed Description

: Functions used in main

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

2.1.2 Function Documentation

2.1.2.1 GPIO_Init()

```
void GPIO_Init (
            void  )
```

: Initialize the GPIO

: It depends on each proyect

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 35 of file Aplication.c.

```
35                      {
36     /*
37      GPIO_SetDIR(LedRED, OUTPUT);
38      GPIO_SetDIR(LedGREEN, OUTPUT);
39      GPIO_SetDIR(LedBLUE, OUTPUT);
40      GPIO_SetDIR(UserKEY, INPUT);
41
42      GPIO_SetPIN(LedRED, LED_OFF);
43      GPIO_SetPIN(LedGREEN, LED_OFF);
44      GPIO_SetPIN(LedBLUE, LED_OFF);
45      */
46     GPIO_SetDIR(RED_LIGTH, OUTPUT);
47     GPIO_SetDIR(YELLOW_LIGTH, OUTPUT);
48     GPIO_SetDIR(GREEN_LIGTH, OUTPUT);
49
50     GPIO_SetPIN(RED_LIGTH, LOW);
51     GPIO_SetPIN(YELLOW_LIGTH, LOW);
52     GPIO_SetPIN(GREEN_LIGTH, LOW);
53 }
```

2.1.2.2 LPC_Init()

```
void LPC_Init (
              void  )
```

: Initialize the board

: It depends on each proyect

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 19 of file Aplication.c.

```
19                      {
20     GPIO_Enable();
21     BoardClockRUN();
22     SysTick_Init();
23     GPIO_Init();
24 }
```

2.1.2.3 Semaphore()

```
void Semaphore (
            void  )
```

: Control of the ligths

: Finite state machine

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

< Reset condition

< Reset condition

< Reset condition

Definition at line 65 of file Aplication.c.

```
65                      {
66     static uint8_t state = RESET;
67     switch (state) {
68     case RESET:
69         GPIO_SetPIN(RED_LIGTH, LOW);
70         GPIO_SetPIN(YELLOW_LIGTH, LOW);
71         GPIO_SetPIN(GREEN_LIGTH, LOW);
72         tick = R_TICK;
73         GPIO_SetPIN(RED_LIGTH, HIGH);
74         state = R_STAGE;
75         break;
76     case R_STAGE:
77         if (tick == 0) {
78             tick = RY_TICK;
79             GPIO_SetPIN(YELLOW_LIGTH, HIGH);
80             state = RY_STAGE;
81         }
82         break;
83     case RY_STAGE:
84         if (tick == 0) {
85             tick = G_TICK;
86             GPIO_SetPIN(RED_LIGTH, LOW);
87             GPIO_SetPIN(YELLOW_LIGTH, LOW);
88             GPIO_SetPIN(GREEN_LIGTH, HIGH);
89             state = G_STAGE;
90         }
91         break;
92     case G_STAGE:
93         if (tick == 0) {
94             tick = Y_TICK;
95             GPIO_SetPIN(GREEN_LIGTH, LOW);
96             GPIO_SetPIN(YELLOW_LIGTH, HIGH);
97             state = Y_STAGE;
```

```
98          }
99          break;
100     case Y_STAGE:
101         if (tick == 0) {
102             tick = R_TICK;
103             GPIO_SetPIN(YELLOW_LIGTH, LOW);
104             GPIO_SetPIN(RED_LIGTH, HIGH);
105             state = R_STAGE;
106         }
107         break;
108
109     }
110
111 }
```

2.2 inc/GPIO_FW.h File Reference

: Firmware functions for GPIO

Macros

- #define **PORT0** 0
- #define **PORT1** 1
- #define LedGREEN PORT1 , 0

  Led green in board.
- #define LedBLUE PORT1 , 1

  Led blue in board.
- #define LedRED PORT1 , 2

  Led red in board.
- #define UserKEY PORT0 , 4

  Key in board.
- #define **INPUT** 0
- #define **OUTPUT** 1
- #define **LOW** 0
- #define **HIGH** 1
- #define **ACT_HIGH** 1
- #define **ACT_LOW** 0
- #define LED_ON 0

  The led are active low.
- #define LED_OFF 1

  The led are active low.
- #define BOUNCE 10

  Times to check the bounce.
- #define **SYSAHBCLKCTRL** ((__RW uint32_t ∗) 0x40048080UL)
- #define **SYSAHBCLKCTRL0** SYSAHBCLKCTRL[0]
- #define **SYSAHBCLKCTRL1** SYSAHBCLKCTRL[1]
- #define **GPIO_PBYTE** ((__RW uint8_t ∗) 0xA0000000UL)
- #define **GPIO_PWORD** ((__RW uint32_t ∗) 0xA0001000UL)
- #define **GPIO_DIRP** ((__RW uint32_t ∗) 0xA0002000UL)
- #define **GPIO_PORT** ((__RW uint32_t ∗) 0xA0002100UL)
- #define **GPIO_SETP** ((__RW uint32_t ∗) 0xA0002200UL)
- #define **GPIO_CLRP** ((__RW uint32_t ∗) 0xA0002280UL)
- #define **GPIO_NOTP** ((__RW uint32_t ∗) 0xA0002300UL)
- #define **NO_PULL_UP_DOWN** 0x00
- #define **PULL_DOWN** 0x01

- #define **PULL_UP** 0x02
- #define **REPEATER** 0x03
- #define **HYS_EN** 0x01
- #define **HYS_DIS** 0x00
- #define **INV_INPUT** 0x01
- #define **NOT_INV_INPUT** 0x00
- #define **OD_EN** 0x01
- #define **OD_DIS** 0x00
- #define **BYPASS_FILTER** 0x00
- #define **CLK1_FILTER** 0x01
- #define **CLK2_FILTER** 0x02
- #define **CLK3_FILTER** 0x03
- #define **IOCONCLKDIV0** 0x00
- #define **IOCONCLKDIV1** 0x01
- #define **IOCONCLKDIV2** 0x02
- #define **IOCONCLKDIV3** 0x03
- #define **IOCONCLKDIV4** 0x04
- #define **IOCONCLKDIV5** 0x05
- #define **IOCONCLKDIV6** 0x06
- #define **DAC_EN** 0x01
- #define **DAC_DIS** 0x00
- #define **STD_MODE** 0x00
- #define **STD_GPIO** 0x01
- #define **FAST_MODE** 0x02
- #define **IOCON_** ((__RW uint32_t ∗) 0x40044000UL)

Functions

- void GPIO_Enable (void)

  : Enable GPIO0 and GPIO1
- void GPIO_Disable (void)

  : Disable GPIO0 and GPIO1
- void GPIO_SetDIR (uint8_t port, uint8_t pin, uint8_t dir)

  : Choose GPIO as Input/Output
- void GPIO_SetPIN (uint8_t port, uint8_t pin, uint8_t state)

  : Choose GPIO's output state
- uint8_t GPIO_GetPIN (uint8_t port, uint8_t pin, uint8_t state)

  : Return GPIO's input state
- void GPIO_SetOUT (uint8_t port, uint8_t pin)

  : Put GPIO's out to 1
- void GPIO_ClearOUT (uint8_t port, uint8_t pin)

  : Put GPIO's out to 0
- void GPIO_ToogleOUT (uint8_t port, uint8_t pin)

  : Invert GPIO's out
- void GPIO_DebounceUserKEY (void)

  : Firmware debounce for user key in board
- void GPIO_Debounce (uint8_t port, uint8_t pin, uint8_t state)

  : Firmware debounce for a GPIO
- void IOCONEnable (void)

  : Enable IOCON
- void IOCONDisable (void)

  : Disable IOCON

- uint8_t GetOFFSET (uint8_t port, uint8_t pin)
  : Usefull for SetMode functions
- void GPIO_SetModeINPUT (uint8_t port, uint8_t pin, uint8_t mode)
  : on-chip pull-up/pull-down resistor
- void GPIO_SetModeHYS (uint8_t port, uint8_t pin, uint8_t mode)
  : Hysteresis
- void GPIO_SetModeINV (uint8_t port, uint8_t pin, uint8_t mode)
  : Invert input
- void GPIO_SetModeOD (uint8_t port, uint8_t pin, uint8_t mode)
  : Open drain
- void GPIO_SetModeFILTER (uint8_t port, uint8_t pin, uint8_t mode)
  : Digital filter sample mode
- void GPIO_SetModeCLKDIV (uint8_t port, uint8_t pin, uint8_t mode)
  : Select peripheral clock divider for input filter sampling clock
- void GPIO_SetModeDAC (uint8_t port, uint8_t pin, uint8_t mode)
  : Selects DAC mode
- void GPIO_SetModeI2C (uint8_t port, uint8_t pin, uint8_t mode)
  : Selects I2C mode

2.2.1 Detailed Description

: Firmware functions for GPIO

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

2.2.2 Function Documentation

2.2.2.1 GetOFFSET()

```
uint8_t GetOFFSET (
            uint8_t port,
            uint8_t pin )
```

: Usefull for SetMode functions

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31

Returns

: void

Definition at line 231 of file GPIO_FW.c.

```
231                                                 {
232     uint8_t index;
233     index = port * 32 + pin;
234     return ((offset[index]) / 4);
235 }
```

2.2.2.2 GPIO_ClearOUT()

```
void GPIO_ClearOUT (
             uint8_t port,
             uint8_t pin )
```

: Put GPIO's out to 0

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31

Returns

: void

Definition at line 113 of file GPIO_FW.c.

```
113                                                 {
114     GPIO_CLRP[port] |= (1 « pin);
115 }
```

2.2.2.3 GPIO_Debounce()

```
void GPIO_Debounce (
            uint8_t port,
            uint8_t pin,
            uint8_t state )
```

: Firmware debounce for a GPIO

: Use in SysTick_Handler or in some timer interrupt

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31
	[in] uint8_t state : ACT_LOW,ACT_HIGH

Returns

: void

Definition at line 169 of file GPIO_FW.c.

```
169                                                                 {
170     static uint8_t q = 0;   //Quantity of bounces
171     uint8_t j = 0;          //It captures changes
172
173     if (GPIO_GetPIN(port, pin, state))   // The key is pushed?
174         j = 0x01;               //Something is happening, the key is been pushed
175
176     if (buff_In ^ j) {          // If the key is pushed while q != BOUNCE
177         q++;                        // I change the buffer
178         if (q == BOUNCE) {
179             q = 0;
180             buff_In ^= 0x01;
181         }
182     } else
183         q = 0;
184 }
```

2.2.2.4 GPIO_DebounceUserKEY()

```
void GPIO_DebounceUserKEY (
            void  )
```

: Firmware debounce for user key in board

: Use in SysTick_Handler or in some timer interrupt

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in]

Returns

: void

Definition at line 141 of file GPIO_FW.c.

```
141                                 {
142     static uint8_t q = 0;   //Quantity of bounces
143     uint8_t j = 0;          //It captures changes
144 
145     if (GPIO_GetPIN(UserKEY, ACT_LOW))    // The key is pushed?
146         j = 0x01;               //Something is happening, the key is been pushed
147 
148     if (buff_UserKEY ^ j) {         // If the key is pushed while q != BOUNCE
149         q++;                        // I change the buffer
150         if (q == BOUNCE) {
151             q = 0;
152             buff_UserKEY ^= 0x01;
153         }
154     } else
155         q = 0;
156 }
```

2.2.2.5 GPIO_Disable()

```
void GPIO_Disable (
              void  )
```

: Disable GPIO0 and GPIO1

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 32 of file GPIO_FW.c.

```
32                          {
33     SYSAHBCLKCTRL0&= (~(1«6));
34     SYSAHBCLKCTRL0 &= (~(1«20));
35 }
```

2.2.2.6 GPIO_Enable()

```
void GPIO_Enable (
             void  )
```

: Enable GPIO0 and GPIO1

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 19 of file GPIO_FW.c.

```
19                          {
20     SYSAHBCLKCTRL0|= (1«6);
21     SYSAHBCLKCTRL0 |= (1«20);
22 }
```

2.2.2.7 GPIO_GetPIN()

```
uint8_t GPIO_GetPIN (
             uint8_t port,
             uint8_t pin,
             uint8_t state )
```

: Return GPIO's input state

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31
	[in] uint8_t STATE : ACT_LOW,ACT_HIGH

Returns

: uint8_t : 1 pin == [state] , 0 pin != [state]

Definition at line 81 of file GPIO_FW.c.

```
81                                                                  {
82     port = port * 32 + pin;
83     if ( GPIO_PBYTE[port] == state)
84         return 1;
85     else
86         return 0;
87 }
```

2.2.2.8 GPIO_SetDIR()

```
void GPIO_SetDIR (
              uint8_t port,
              uint8_t pin,
              uint8_t dir )
```

: Choose GPIO as Input/Output

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31
	[in] uint8_t dir : INPUT,OUTPUT

Returns

: void

Definition at line 48 of file GPIO_FW.c.

```
48                                                          {
49     GPIO_DIRP[port] &= (~(1 « pin));
50     GPIO_DIRP[port] |= (dir « pin);
51 }
```

2.2.2.9 GPIO_SetModeCLKDIV()

```
void GPIO_SetModeCLKDIV (
            uint8_t port,
            uint8_t pin,
            uint8_t mode )
```

: Select peripheral clock divider for input filter sampling clock

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port: PORT0,PORT1 : [in] uint8_t pin: 0,31 : [in] uint8_t mode: IOCONCLKDIV0 to IOCONCLKDIV6

Returns

: void

Definition at line 338 of file GPIO_FW.c.

```
338                                                                 {
339     uint8_t offset;
340     offset = GetOFFSET(port, pin);
341     IOCON_[offset] &= (~(0x07 « 13));
342     IOCON_[offset] |= (mode « 13);
343 }
```

2.2.2.10 GPIO_SetModeDAC()

```
void GPIO_SetModeDAC (
            uint8_t port,
            uint8_t pin,
            uint8_t mode )
```

: Selects DAC mode

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port: PORT0,PORT1 : [in] uint8_t pin: 0,31 : [in] uint8_t mode: DAC_EN,DAC_DIS

Returns

: void

Definition at line 356 of file GPIO_FW.c.

```
356                                                                 {
357     uint8_t offset;
358     offset = GetOFFSET(port, pin);
359     IOCON_[offset] &= (~(0x01 « 16));
360     IOCON_[offset] |= (mode « 16);
361 }
```

2.2.2.11 GPIO_SetModeFILTER()

```
void GPIO_SetModeFILTER (
             uint8_t port,
             uint8_t pin,
             uint8_t mode )
```

: Digital filter sample mode

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port: PORT0,PORT1 : [in] uint8_t pin: 0,31 : [in] uint8_t mode: BYPASS_FILTER,CLK1_FILTER,CLK2_FILTER,CLK3_FILTER

Returns

: void

Definition at line 320 of file GPIO_FW.c.

```
320                                                                 {
321     uint8_t offset;
322     offset = GetOFFSET(port, pin);
323     IOCON_[offset] &= (~(0x03 « 11));
324     IOCON_[offset] |= (mode « 11);
325 }
```

2.2.2.12 GPIO_SetModeHYS()

```
void GPIO_SetModeHYS (
            uint8_t port,
            uint8_t pin,
            uint8_t mode )
```

: Hysteresis

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port: PORT0,PORT1 : [in] uint8_t pin: 0,31 : [in] uint8_t mode:HYS_EN,HYS_DIS

Returns

: void

Definition at line 266 of file GPIO_FW.c.

```
266                                                                 {
267     uint8_t offset;
268     offset = GetOFFSET(port, pin);
269     IOCON_[offset] &= (~(0x01 « 5));
270     IOCON_[offset] |= (mode « 5);
271 }
```

2.2.2.13 GPIO_SetModeI2C()

```
void GPIO_SetModeI2C (
            uint8_t port,
```

```
            uint8_t pin,
            uint8_t mode )
```

: Selects I2C mode

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port: PORT0,PORT1 : [in] uint8_t pin: 0,31 : [in] uint8_t mode:STD_MODE,STD_GPIO,FAST_MODE

Returns

: void

Definition at line 374 of file GPIO_FW.c.

```
374                                                     {
375     uint8_t offset;
376     offset = GetOFFSET(port, pin);
377     IOCON_[offset] &= (~(0x03 « 8));
378     IOCON_[offset] |= (mode « 8);
379 }
```

2.2.2.14 GPIO_SetModeINPUT()

```
void GPIO_SetModeINPUT (
            uint8_t port,
            uint8_t pin,
            uint8_t mode )
```

: on-chip pull-up/pull-down resistor

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port: PORT0,PORT1 : [in] uint8_t pin: 0,31 : [in] uint8_t mode:NO_PULL_UP_DOWN,PULL_DOWN,PULL_UP,REPEATER

Returns

: void

Definition at line 248 of file GPIO_FW.c.

```
248                                                                 {
249     uint8_t offset;
250     offset = GetOFFSET(port, pin);
251     IOCON_[offset] &= (~(0x03 « 3));
252     IOCON_[offset] |= (mode « 3);
253 }
```

2.2.2.15 GPIO_SetModeINV()

```
void GPIO_SetModeINV (
             uint8_t port,
             uint8_t pin,
             uint8_t mode )
```

: Invert input

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port: PORT0,PORT1 : [in] uint8_t pin: 0,31 : [in] uint8_t mode: INV_INPUT,NOT_INV_INPUT

Returns

: void

Definition at line 284 of file GPIO_FW.c.

```
284                                                                 {
285     uint8_t offset;
286     offset = GetOFFSET(port, pin);
287     IOCON_[offset] &= (~(0x01 « 6));
288     IOCON_[offset] |= (mode « 6);
289 }
```

2.2.2.16 GPIO_SetModeOD()

```
void GPIO_SetModeOD (
            uint8_t port,
            uint8_t pin,
            uint8_t mode )
```

: Open drain

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port: PORT0,PORT1 : [in] uint8_t pin: 0,31 : [in] uint8_t mode: OD_EN,OD_DIS

Returns

: void

Definition at line 302 of file GPIO_FW.c.

```
302                                                                 {
303     uint8_t offset;
304     offset = GetOFFSET(port, pin);
305     IOCON_[offset] &= (~(0x01 « 10));
306     IOCON_[offset] |= (mode « 10);
307 }
```

2.2.2.17 GPIO_SetOUT()

```
void GPIO_SetOUT (
            uint8_t port,
            uint8_t pin )
```

: Put GPIO's out to 1

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31

Returns

: void

Definition at line 99 of file GPIO_FW.c.

```
99                                              {
100     GPIO_SETP[port] |= (1 « pin);
101 }
```

2.2.2.18 GPIO_SetPIN()

```
void GPIO_SetPIN (
             uint8_t port,
             uint8_t pin,
             uint8_t state )
```

: Choose GPIO's output state

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31
	[in] uint8_t state : LOW,HIGH

Returns

: void

Definition at line 64 of file GPIO_FW.c.

```
64                                                      {
65     port = port * 32 + pin;
66     GPIO_PBYTE[port] &= (~1);
67     GPIO_PBYTE[port] |= state;
68 }
```

2.2.2.19 GPIO_ToogleOUT()

```
void GPIO_ToogleOUT (
             uint8_t port,
             uint8_t pin )
```

: Invert GPIO's out

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31

Returns

: void

Definition at line 127 of file GPIO_FW.c.

```
127                                                 {
128     GPIO_NOTP[port] |= (1 « pin);
129 }
```

2.2.2.20 IOCONDisable()

```
void IOCONDisable (
             void  )
```

: Disable IOCON

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in]

Returns

: void

Definition at line 208 of file GPIO_FW.c.

```
208                         {
209     SYSAHBCLKCTRL0&= (~(1<<18));
210 }
```

2.2.2.21 IOCONEnable()

```
void IOCONEnable (
             void  )
```

: Enable IOCON

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in]

Returns

: void

Definition at line 195 of file GPIO_FW.c.

```
195                         {
196     SYSAHBCLKCTRL0|= (1<<18);
197 }
```

2.3 inc/GPIO_SW.h File Reference

: Software functions for GPIO

Functions

- uint8_t GetUserKEY (void)

  : State of the user key in board
- uint8_t GetInput (void)

  : State of the input

2.3.1 Detailed Description

: Software functions for GPIO

: These are functions in a higher layer of abstraction

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

2.3.2 Function Documentation

2.3.2.1 GetInput()

```
uint8_t GetInput (
            void  )
```

: State of the input

: Is necessary using GPIO_Debounce

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: uint8_t 1 if input pressed, 0 if input pressed

Definition at line 48 of file GPIO_SW.c.

```
48                          {
49     static uint8_t buff_before = 0x00;
50
51     if ( buff_In == 0x01 && buff_before == 0x00 ){
52         buff_before = 0x01;
53         return (1);
54     }
55     else if ( buff_In == 0x01 && buff_before == 0x01 )
56         return (0);
57     else if ( buff_In == 0x00 && buff_before == 0x01 )
58         return (0);
59     else
60         return (0);
61 }
```

2.3.2.2 GetUserKEY()

```
uint8_t GetUserKEY (
             void  )
```

: State of the user key in board

: Is necessary using GPIO_DebounceUserKEY

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: uint8_t 1 if user key pressed, 0 if user key not

Definition at line 21 of file GPIO_SW.c.

```
21                           {
22     static uint8_t buff_before = 0x00;
23
24     if ( buff_UserKEY == 0x01 && buff_before == 0x00 ){
25         buff_before = 0x01;
26         return (1);
27     }
28     else if ( buff_UserKEY == 0x01 && buff_before == 0x01 )
29         return (0);
30     else if ( buff_UserKEY == 0x00 && buff_before == 0x01 ){
31         buff_before = 0x00;
32         return (0);
33     }
34     else
35         return (0);
36 }
```

2.4 inc/LPC845.h File Reference

: Declarations for type of data

Macros

- #define **__R** volatile const
- #define **__W** volatile
- #define **__RW** volatile

Typedefs

- typedef unsigned int **uint32_t**
- typedef unsigned short **uint16_t**
- typedef unsigned char **uint8_t**

2.4.1 Detailed Description

: Declarations for type of data

: Only contains macros

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

2.5 inc/SwitchMatrix_FW.h File Reference

: Firmware functions for SWM

Macros

- #define **PINASSIGN** ((__RW uint32_t ∗) 0x4000C000UL)
- #define **PINENABLE** ((__RW uint32_t ∗) 0x4000C1C0UL)

Enumerations

- enum { **BYTE0** , **BYTE1** , **BYTE2** , **BYTE3** }
- enum {
  UO_TXD , **UO_SCLK** , **U1_CTS** , **U2_RTS** ,
  SPI0_MOSI , **SPI0_SSEL2** , **SPI1_MISO** , **SCT_IN1** ,
  SCT_OUT1 , **SCT_OUT5** , **I2C2_SDA** , **COMP0_OUT** ,
  UART3_RXD , **UART4_SCLK** , **T0_MAT3** }
- enum {
  U0_RXD , **U1_TXD** , **U0_SCLK** , **U2_CTS** ,
  SPI0_MISO , **SPI0_SSEL3** , **SPI1_SSEL0** , **SCT_IN2** ,
  SCT_OUT2 , **SCT_OUT6** , **I2C2_SCL** , **CLKOUT** ,
  UART3_SCLK , **T0_MAT0** , **T0_CAP0** }
- enum {
  UO_RTS , **U1_RXD** , **U2_TXD** , **U2_SCLK** ,
  SPI0_SSEL0 , **SPI1_SCK** , **SPI1_SSEL1** , **SCT_IN3** ,
  SCT_OUT3 , **I2C1_SDA** , **I2C3_SDA** , **GPIO_INT_BMAT** ,
  UART4_TXD , **T0_MAT1** , **T0_CAP1** }
- enum {
  UO_CTS , **U1_RTS** , **UO_RXD** , **SPIO_SCK** ,
  SPI0_SSEL1 , **SPI1_MOSI** , **SCT0_IN0** , **SCT_OUT0** ,
  SCT_OUT4 , **I2C1_SCL** , **I2C3_SCL** , **UART3_TXD** ,
  UART4_RXD , **T0_MAT2** , **T0_CAP2** }
- enum {
  ADC_0 , **ADC_1** , **ADC_2** , **ADC_3** ,
  ADC_4 , **ADC_5** , **ADC_6** , **ADC_7** ,
  ADC_8 , **ADC_9** , **ADC_10** , **ADC_11** ,
  DACOUT0 , **DACOUT1** , **CAPT_X0** , **CAPT_X1** ,
  CAPT_X2 , **CAPT_X3** }
- enum {
  CAPT_X4 , **CAPT_X5** , **CAPT_X6** , **CAPT_X7** ,
  CAPT_X8 , **CAPT_YL** , **CAPT_YH** }

Functions

- void SWM (uint8_t port, uint8_t pin, uint8_t assign, uint8_t byte)
  : Assign movable functions for pin
- void SWM_PinEnable (uint8_t port, uint8_t pin, uint8_t ena)
  : Enable pin works as value passed in ena
- void SWM_Enable (void)
  : Enable SWM
- void SWM_Disable (void)
  : Disable SWM

2.5.1 Detailed Description

: Firmware functions for SWM

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

2.5.2 Enumeration Type Documentation

2.5.2.1 anonymous enum

```
anonymous enum
```

Enumerator

UO_TXD	Possible assign.

Definition at line 38 of file SwitchMatrix_FW.h.

```
38      {
39     UO_TXD,
40     UO_SCLK,
41     U1_CTS,
42     U2_RTS,
43     SPI0_MOSI,
44     SPI0_SSEL2,
45     SPI1_MISO,
46     SCT_IN1,
47     SCT_OUT1,
48     SCT_OUT5,
49     I2C2_SDA,
50     COMP0_OUT,
51     UART3_RXD,
52     UART4_SCLK,
53     T0_MAT3
54 };
```

2.5.2.2 anonymous enum

```
anonymous enum
```

Enumerator

U0_RXD	Possible assign.

Definition at line 56 of file SwitchMatrix_FW.h.

```
56      {
57     U0_RXD,
58     U1_TXD,
59     U0_SCLK,
60     U2_CTS,
61     SPI0_MISO,
62     SPI0_SSEL3,
63     SPI1_SSEL0,
64     SCT_IN2,
65     SCT_OUT2,
66     SCT_OUT6,
67     I2C2_SCL,
68     CLKOUT,
69     UART3_SCLK,
70     T0_MAT0,
71     T0_CAP0
72 };
```

2.5.2.3 anonymous enum

anonymous enum

Enumerator

UO_RTS	Possible assign.

Definition at line 74 of file SwitchMatrix_FW.h.

```
74      {
75      UO_RTS,
76      U1_RXD,
77      U2_TXD,
78      U2_SCLK,
79      SPI0_SSEL0,
80      SPI1_SCK,
81      SPI1_SSEL1,
82      SCT_IN3,
83      SCT_OUT3,
84      I2C1_SDA,
85      I2C3_SDA,
86      GPIO_INT_BMAT,
87      UART4_TXD,
88      T0_MAT1,
89      T0_CAP1
90  };
```

2.5.2.4 anonymous enum

anonymous enum

Enumerator

UO_CTS	Possible assign.

Definition at line 92 of file SwitchMatrix_FW.h.

```
92      {
93      UO_CTS,
94      U1_RTS,
95      UO_RXD,
96      SPIO_SCK,
97      SPI0_SSEL1,
98      SPI1_MOSI,
99      SCT0_IN0,
100      SCT_OUT0,
101      SCT_OUT4,
102      I2C1_SCL,
103      I2C3_SCL,
104      UART3_TXD,
105      UART4_RXD,
106      T0_MAT2,
107      T0_CAP2
108  };
```

2.5.3 Function Documentation

2.5.3.1 SWM()

```
void SWM (
              uint8_t port,
              uint8_t pin,
              uint8_t assign,
              uint8_t byte )
```

: Assign movable functions for pin

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31
	[in] uint8_t assign :
	[in] uint8_t byte : BYTE0,BYTE1,BYTE2,BYTE3

Returns

: void

Definition at line 22 of file SwitchMatrix_FW.c.

```
22                                                                  {
23     pin = pin + 0x20 * port; //PIO0[0:31] 0x00 to 0x1F PIO1[0:21] 0x1F to 0x35
24     PINASSIGN[assign] |= (pin « byte);
25 }
```

2.5.3.2 SWM_Disable()

```
void SWM_Disable (
              void  )
```

: Disable SWM

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 67 of file SwitchMatrix_FW.c.

```
67                          {
68     SYSAHBCLKCTRL0&= (~(1«7));
69 }
```

2.5.3.3 SWM_Enable()

```
void SWM_Enable (
             void  )
```

: Enable SWM

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 54 of file SwitchMatrix_FW.c.

```
54                          {
55     SYSAHBCLKCTRL0|= (1«7);
56 }
```

2.5.3.4 SWM_PinEnable()

```
void SWM_PinEnable (
             uint8_t port,
```

```
            uint8_t pin,
            uint8_t ena )
```

: Enable pin works as value passed in ena

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31
	[in] uint8_t ena : READ Page 143 UserManual. There are multiple choices

Returns

: void

Definition at line 38 of file SwitchMatrix_FW.c.

```
38                                                          {
39     if (port == PORT1)       //PIENABLE[0] -> PIO0_0 .... PIO1_2
40         if (pin < 3)         //PIENABLE10] -> PIO1_3 .... PIO1_21
41             port = PORT0;
42     PINENABLE[port] |= (1 << ena);
43 }
```

2.6 inc/SYSCON_FW.h File Reference

: Firmware functions for SYSCON

Macros

- #define **SYSCON_ADD** ((__RW uint32_t *) 0x40048000UL)
- #define **SYSMEMREMAP** SYSCON_ADD [0]
- #define **SYSPLLCTRL** SYSCON_ADD [2]
- #define **SYSPLLSTAT** SYSCON_ADD [3]
- #define **SYSOSCCTRL** SYSCON_ADD [8]
- #define **WDTOSCCTRL** SYSCON_ADD [9]
- #define **FROOSCCTRL** SYSCON_ADD [10]
- #define **FRODIRECTCLKUEN** SYSCON_ADD [12]
- #define **SYSRSTSTAT** SYSCON_ADD [14]
- #define **SYSPLLCLKSEL** SYSCON_ADD [16]
- #define **SYSPLLCLKUEN** SYSCON_ADD [17]
- #define **MAINCLKPLLSEL** SYSCON_ADD [18]

- #define **MAINCLKPLLUEN** SYSCON_ADD [19]
- #define **MAINCLKSEL** SYSCON_ADD [20]
- #define **MAINCLKUEN** SYSCON_ADD [21]
- #define **SYSAHBCLKDIV** SYSCON_ADD [22]
- #define **CAPTCLKSEL** SYSCON_ADD [24]
- #define **ADCCLKSEL** SYSCON_ADD [25]
- #define **ADCCLKDIV** SYSCON_ADD [26]
- #define **SCTCLKSEL** SYSCON_ADD [27]
- #define **SCTCLKDIV** SYSCON_ADD [28]
- #define **EXTCLKSEL** SYSCON_ADD [29]
- #define **_SYSAHBCLKCTRL0** SYSCON_ADD [32]
- #define **_SYSAHBCLKCTRL1** SYSCON_ADD [33]
- #define **PRESETCTRL0** SYSCON_ADD [34]
- #define **PRESETCTRL1** SYSCON_ADD [35]
- #define **UART0CLKSEL** SYSCON_ADD [36]
- #define **UART1CLKSEL** SYSCON_ADD [37]
- #define **UART2CLKSEL** SYSCON_ADD [38]
- #define **UART3CLKSEL** SYSCON_ADD [39]
- #define **UART4CLKSEL** SYSCON_ADD [40]
- #define **I2C0CLKSEL** SYSCON_ADD [41]
- #define **I2C1CLKSEL** SYSCON_ADD [42]
- #define **I2C2CLKSEL** SYSCON_ADD [43]
- #define **I2C3CLKSEL** SYSCON_ADD [44]
- #define **SPI0CLKSEL** SYSCON_ADD [45]
- #define **SPI1CLKSEL** SYSCON_ADD [46]
- #define **FRG0DIV** SYSCON_ADD [52]
- #define **FRG0MULT** SYSCON_ADD [53]
- #define **FRG0CLKSEL** SYSCON_ADD [54]
- #define **FRG1DIV** SYSCON_ADD [56]
- #define **FRG1MULT** SYSCON_ADD [57]
- #define **FRG1CLKSEL** SYSCON_ADD [58]
- #define **CLKOUTSEL** SYSCON_ADD [60]
- #define **CLKOUTDIV** SYSCON_ADD [61]
- #define **EXTTRACECMD** SYSCON_ADD [63]
- #define **PIOPORCAP0** SYSCON_ADD [64]
- #define **PIOPORCAP1** SYSCON_ADD [65]
- #define **_IOCONCLKDIV6** SYSCON_ADD [77]
- #define **_IOCONCLKDIV5** SYSCON_ADD [78]
- #define **_IOCONCLKDIV4** SYSCON_ADD [79]
- #define **_IOCONCLKDIV3** SYSCON_ADD [80]
- #define **_IOCONCLKDIV2** SYSCON_ADD [81]
- #define **_IOCONCLKDIV1** SYSCON_ADD [82]
- #define **_IOCONCLKDIV0** SYSCON_ADD [83]
- #define **BODCTRL** SYSCON_ADD [84]
- #define **SYSTCKCAL** SYSCON_ADD [85]
- #define **IRQLATENCY** SYSCON_ADD [92]
- #define **NMISRC** SYSCON_ADD [93]
- #define **PINTSEL0** SYSCON_ADD [94]
- #define **PINTSEL1** SYSCON_ADD [95]
- #define **PINTSEL2** SYSCON_ADD [96]
- #define **PINTSEL3** SYSCON_ADD [97]
- #define **PINTSEL4** SYSCON_ADD [98]
- #define **PINTSEL5** SYSCON_ADD [99]
- #define **PINTSEL6** SYSCON_ADD [100]
- #define **PINTSEL7** SYSCON_ADD [101]

- #define **STARTERP0** SYSCON_ADD [129]
- #define **STARTERP1** SYSCON_ADD [133]
- #define **PDSLEEPCFG** SYSCON_ADD [140]
- #define **PDAWAKECFG** SYSCON_ADD [141]
- #define **PDRUNCFG** SYSCON_ADD [142]
- #define **DEVICE_ID** SYSCON_ADD [254]
- #define **CLOCK_FRO_SETTING_API_ROM_ADDRESS** 0x0F0026F5U
- #define **F30MHz** 30000U
- #define **FRO_OUT_PowerDown** 1
- #define **FRO_PD** 2
- #define **SYSCON_FROOSCCTRL_FRO_DIRECT_MASK** (0x20000U)
- #define **SYSCON_FROOSCCTRL_FRO_DIRECT_SHIFT** (17U)
- #define **kCLOCK_FroSrcFroOsc** 1U << SYSCON_FROOSCCTRL_FRO_DIRECT_SHIFT
- #define **kPDRUNCFG_PD_SYSOSC** 0x20
- #define **CLK_FROM_SYS_OSC** 0x00
- #define **FREQ30MHz** 30000000U
- #define **CLK_SYS_PLLSRCFRODIV** 0x03
- #define **CLOCK_FAIM_BASE** 0x50010000U
- #define **SYSPLL_MIN_FCCO_FREQ_HZ** 156000000U
- #define **SYSCON_SYSPLLCTRL_MSEL_MASK** 0x1FU
- #define **SYSCON_SYSPLLCTRL_MSEL_SHIFT** (0U)
- #define **SYSCON_SYSPLLCTRL_PSEL_MASK** 0x60U
- #define **SYSCON_SYSPLLCTRL_PSEL_SHIFT** (5U)
- #define **SYSCON_SYSPLLCTRL_MSEL**(x) (((uint32_t)(((uint32_t)(x)) << SYSCON_SYSPLLCTRL_↩ MSEL_SHIFT)) & SYSCON_SYSPLLCTRL_MSEL_MASK)
- #define **SYSCON_SYSPLLCTRL_PSEL**(x) (((uint32_t)(((uint32_t)(x)) << SYSCON_SYSPLLCTRL_↩ PSEL_SHIFT)) & SYSCON_SYSPLLCTRL_PSEL_MASK)
- #define **CLK_MAIN_CLK_MUX_GET_MUX**(x) ((uint32_t)(x) & 0xFFU)
- #define **CLK_MAIN_CLK_MUX_GET_PRE_MUX**(x) (((uint32_t)(x) >> 8U) & 0xFFU)
- #define **SYSCON_MAINCLKSEL_SEL_MASK** 0x03U
- #define **SYSCON_MAINCLKSEL_SEL_SHIFT** (0U)
- #define **SYSCON_MAINCLKSEL_SEL**(x) (((uint32_t)(((uint32_t)(x)) << SYSCON_MAINCLKSEL_SEL_↩ SHIFT)) & SYSCON_MAINCLKSEL_SEL_MASK)
- #define **SYSCON_MAINCLKPLLSEL_SEL_MASK** (0x3U)
- #define **SYSCON_MAINCLKPLLSEL_SEL_SHIFT** (0U)
- #define **SYSCON_MAINCLKPLLSEL_SEL**(x) (((uint32_t)(((uint32_t)(x)) << SYSCON_MAINCLKPLLSEL↩ _SEL_SHIFT)) & SYSCON_MAINCLKPLLSEL_SEL_MASK)
- #define **kCLOCK_MainClkSrcFro** 0
- #define **SYSCON_SYSAHBCLKDIV_DIV**(x) (((uint32_t)(((uint32_t)(x)) << SYSCON_SYSAHBCLKDIV_↩ DIV_SHIFT)) & SYSCON_SYSAHBCLKDIV_DIV_MASK)
- #define **SYSCON_SYSAHBCLKDIV_DIV_MASK** 0xFFU
- #define **SYSCON_SYSAHBCLKDIV_DIV_SHIFT** (0U)

Functions

- void BoardClockRUN ()

  : Runs clock at 30MHz
- void **ClockSetFroOscFREQ** (uint32_t freq)
- void **PowerDisablePD** (uint8_t en)
- void **CLOCK_SetFroOutClkSrc** (uint32_t src)
- void **CLOCK_Select** (uint8_t sel)
- void **CLOCK_InitSystemPll** (uint32_t freq, uint8_t src)
- uint32_t **CLOCK_GetSystemPLLInClockRate** (void)
- uint32_t **CLOCK_GetFroFreq** (void)
- uint32_t **FindSyestemPllPsel** (uint32_t outFreq)
- void **CLOCK_SetMainClkSrc** (uint32_t src)
- void **CLOCK_SetCoreSysClkDiv** (uint32_t value)

2.6.1 Detailed Description

: Firmware functions for SYSCON

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

2.6.2 Function Documentation

2.6.2.1 BoardClockRUN()

```
void BoardClockRUN (
             void  )
```

: Runs clock at 30MHz

: Select clock from fro

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 19 of file SYSCON_FW.c.

```
19                          {
20     PowerDisablePD(FRO_OUT_PowerDown);
21     PowerDisablePD(FRO_PD);
22     ClockSetFroOscFREQ(F30MHz);
23     CLOCK_SetFroOutClkSrc(kCLOCK_FroSrcFroOsc);
24     PowerDisablePD(kPDRUNCFG_PD_SYSOSC);
25     CLOCK_Select(CLK_FROM_SYS_OSC);
26     CLOCK_InitSystemPll(FREQ30MHz, CLK_SYS_PLLSRCFRODIV);
27     CLOCK_SetMainClkSrc(kCLOCK_MainClkSrcFro);
28     CLOCK_SetCoreSysClkDiv(1U);
29 }
```

2.7 inc/SysTick_FW.h File Reference

: Firmware functions for SysTick

Macros

- #define TICK_OUT_1S 100

  Systick interrupt each 1 second.
- #define **SysTick_** ((__RW uint32_t ∗) 0xE000E000UL)
- #define **SYST_CSR** SysTick_[4]
- #define **SYST_RVR** SysTick_[5]
- #define **SYST_CVR** SysTick_[6]
- #define **SYST_CALIB** SysTick_[7]
- #define **SYSTICK_ENABLE_INTERRUPT_CLK** 0x07
- #define **SYSTICK_DISABLE** 0x00
- #define **FRE30MHz** 30000U

Functions

- void SysTick_Init (void)

  : Initialize the systick
- void SysTick_Off (void)

  : Stops the systick
- void SysTick_Set (uint32_t freq)

  : Set the counter as freq∗10mS -1

2.7.1 Detailed Description

: Firmware functions for SysTick

: Used for 30 MHz

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

2.7.2 Function Documentation

2.7.2.1 SysTick_Init()

```
void SysTick_Init (
            void  )
```

: Initialize the systick

: Enable SysTick, enable interrupt and set the counter

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 19 of file SysTick_FW.c.

```
19                      {
20     SysTick_Set(FRE30MHz);
21     SYST_CSR = SYSTICK_ENABLE_INTERRUPT_CLK;
22     SYST_CVR = 0;
23 }
```

2.7.2.2 SysTick_Off()

```
void SysTick_Off (
             void  )
```

: Stops the systick

: disable SysTick, disable interrupt

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 34 of file SysTick_FW.c.

```
34                      {
35     SYST_CSR = SYSTICK_DISABLE;
36 }
```

2.7.2.3 SysTick_Set()

```
void SysTick_Set (
             uint32_t freq )
```

: Set the counter as freq∗10mS -1

: Always use at 30MHz

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint32_t freq: FRE30MHz

Returns

: void

Definition at line 47 of file SysTick_FW.c.

```
47                                  {
48     SYST_RVR = freq*10 - 1; // 30MHz*10mS-1
49 }
```

2.8 source/03-Semaphore.c File Reference

: Entry point for the program

```
#include "Aplication.h"
```

Functions

- int main (void)
  : Main Function

Variables

- uint32_t tick = 0
  Var for SysTick_Handler.

2.8.1 Detailed Description

: Entry point for the program

: Semaphore with 4 stages

Author

: Tobias Bavasso Piizzi

Date

: 05/01/2021

2.8.2 Function Documentation

2.8.2.1 main()

```
:int main (
            void  )
```

: Main Function

: initialize the system and stay in the while

Author

: Tobias Bavasso Piizzi

Date

: 05/01/2021

Parameters

	[in] void

Returns

: int

< FSM in aplication.c

Definition at line 21 of file 03-Semaphore.c.

```
21                  {
22     LPC_Init();
23     while(1) {
24         Semaphore();
25     }
26     return 0 ;
27 }
```

2.8.3 Variable Documentation

2.8.3.1 tick

```
uint32_t tick = 0
```

Var for SysTick_Handler.

Declared in main.

Definition at line 20 of file 03-Semaphore.c.

2.9 source/Aplication.c File Reference

: Functions used in main

```
#include "Aplication.h"
```

Functions

- void LPC_Init (void)
  : Initialize the board
- void GPIO_Init (void)
  : Initialize the GPIO
- void Semaphore (void)
  : Control of the ligths

Variables

- uint32_t tick
  Declared in main.

2.9.1 Detailed Description

: Functions used in main

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

2.9.2 Function Documentation

2.9.2.1 GPIO_Init()

```
:void GPIO_Init (
             void  )
```

: Initialize the GPIO

: It depends on each proyect

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 35 of file Aplication.c.

```
35                      {
36     /*
37      GPIO_SetDIR(LedRED, OUTPUT);
38      GPIO_SetDIR(LedGREEN, OUTPUT);
39      GPIO_SetDIR(LedBLUE, OUTPUT);
40      GPIO_SetDIR(UserKEY, INPUT);
41
42      GPIO_SetPIN(LedRED, LED_OFF);
43      GPIO_SetPIN(LedGREEN, LED_OFF);
44      GPIO_SetPIN(LedBLUE, LED_OFF);
45      */
46     GPIO_SetDIR(RED_LIGTH, OUTPUT);
47     GPIO_SetDIR(YELLOW_LIGTH, OUTPUT);
48     GPIO_SetDIR(GREEN_LIGTH, OUTPUT);
49
50     GPIO_SetPIN(RED_LIGTH, LOW);
51     GPIO_SetPIN(YELLOW_LIGTH, LOW);
52     GPIO_SetPIN(GREEN_LIGTH, LOW);
53 }
```

2.9.2.2 LPC_Init()

```
:void LPC_Init (
              void  )
```

: Initialize the board

: It depends on each proyect

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 19 of file Aplication.c.

```
19                      {
20     GPIO_Enable();
21     BoardClockRUN();
22     SysTick_Init();
23     GPIO_Init();
24 }
```

2.9.2.3 Semaphore()

```
:void Semaphore (
             void  )
```

: Control of the ligths

: Finite state machine

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

< Reset condition

< Reset condition

< Reset condition

Definition at line 65 of file Aplication.c.

```
65                      {
66     static uint8_t state = RESET;
67     switch (state) {
68     case RESET:
69         GPIO_SetPIN(RED_LIGTH, LOW);
70         GPIO_SetPIN(YELLOW_LIGTH, LOW);
71         GPIO_SetPIN(GREEN_LIGTH, LOW);
72         tick = R_TICK;
73         GPIO_SetPIN(RED_LIGTH, HIGH);
74         state = R_STAGE;
75         break;
76     case R_STAGE:
77         if (tick == 0) {
78             tick = RY_TICK;
79             GPIO_SetPIN(YELLOW_LIGTH, HIGH);
80             state = RY_STAGE;
81         }
82         break;
83     case RY_STAGE:
84         if (tick == 0) {
```

```
85              tick = G_TICK;
86              GPIO_SetPIN(RED_LIGTH, LOW);
87              GPIO_SetPIN(YELLOW_LIGTH, LOW);
88              GPIO_SetPIN(GREEN_LIGTH, HIGH);
89              state = G_STAGE;
90          }
91          break;
92      case G_STAGE:
93          if (tick == 0) {
94              tick = Y_TICK;
95              GPIO_SetPIN(GREEN_LIGTH, LOW);
96              GPIO_SetPIN(YELLOW_LIGTH, HIGH);
97              state = Y_STAGE;
98          }
99          break;
100     case Y_STAGE:
101         if (tick == 0) {
102             tick = R_TICK;
103             GPIO_SetPIN(YELLOW_LIGTH, LOW);
104             GPIO_SetPIN(RED_LIGTH, HIGH);
105             state = R_STAGE;
106         }
107         break;
108
109     }
110
111 }
```

2.9.3 Variable Documentation

2.9.3.1 tick

```
uint32_t tick  [extern]
```

Declared in main.

Declared in main.

Definition at line 20 of file 03-Semaphore.c.

2.10 source/GPIO_FW.c File Reference

: Firmware functions for GPIO

```
#include "Aplication.h"
```

Functions

- void GPIO_Enable (void)
  : Enable GPIO0 and GPIO1
- void GPIO_Disable (void)
  : Disable GPIO0 and GPIO1
- void GPIO_SetDIR (uint8_t port, uint8_t pin, uint8_t dir)
  : Choose GPIO as Input/Output
- void GPIO_SetPIN (uint8_t port, uint8_t pin, uint8_t state)

: Choose GPIO's output state

- uint8_t GPIO_GetPIN (uint8_t port, uint8_t pin, uint8_t state)

  : Return GPIO's input state
- void GPIO_SetOUT (uint8_t port, uint8_t pin)

  : Put GPIO's out to 1
- void GPIO_ClearOUT (uint8_t port, uint8_t pin)

  : Put GPIO's out to 0
- void GPIO_ToogleOUT (uint8_t port, uint8_t pin)

  : Invert GPIO's out
- void GPIO_DebounceUserKEY (void)

  : Firmware debounce for user key in board
- void GPIO_Debounce (uint8_t port, uint8_t pin, uint8_t state)

  : Firmware debounce for a GPIO
- void IOCONEnable (void)

  : Enable IOCON
- void IOCONDisable (void)

  : Disable IOCON
- uint8_t GetOFFSET (uint8_t port, uint8_t pin)

  : Usefull for SetMode functions
- void GPIO_SetModeINPUT (uint8_t port, uint8_t pin, uint8_t mode)

  : on-chip pull-up/pull-down resistor
- void GPIO_SetModeHYS (uint8_t port, uint8_t pin, uint8_t mode)

  : Hysteresis
- void GPIO_SetModeINV (uint8_t port, uint8_t pin, uint8_t mode)

  : Invert input
- void GPIO_SetModeOD (uint8_t port, uint8_t pin, uint8_t mode)

  : Open drain
- void GPIO_SetModeFILTER (uint8_t port, uint8_t pin, uint8_t mode)

  : Digital filter sample mode
- void GPIO_SetModeCLKDIV (uint8_t port, uint8_t pin, uint8_t mode)

  : Select peripheral clock divider for input filter sampling clock
- void GPIO_SetModeDAC (uint8_t port, uint8_t pin, uint8_t mode)

  : Selects DAC mode
- void GPIO_SetModeI2C (uint8_t port, uint8_t pin, uint8_t mode)

  : Selects I2C mode

Variables

- __RW uint8_t **buff_UserKEY** = 0
- __RW uint8_t **buff_In** = 0
- uint8_t **offset** [ ]

2.10.1 Detailed Description

: Firmware functions for GPIO

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

2.10.2 Function Documentation

2.10.2.1 GetOFFSET()

```
:uint8_t GetOFFSET (
            uint8_t port,
            uint8_t pin )
```

: Usefull for SetMode functions

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31

Returns

: void

Definition at line 231 of file GPIO_FW.c.

```
231                                                 {
232     uint8_t index;
233     index = port * 32 + pin;
234     return ((offset[index]) / 4);
235 }
```

2.10.2.2 GPIO_ClearOUT()

```
:void GPIO_ClearOUT (
            uint8_t port,
            uint8_t pin )
```

: Put GPIO's out to 0

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31

Returns

: void

Definition at line 113 of file GPIO_FW.c.

```
113                                                 {
114     GPIO_CLRP[port] |= (1 « pin);
115 }
```

2.10.2.3 GPIO_Debounce()

```
:void GPIO_Debounce (
             uint8_t port,
             uint8_t pin,
             uint8_t state )
```

: Firmware debounce for a GPIO

: Use in SysTick_Handler or in some timer interrupt

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31
	[in] uint8_t state : ACT_LOW,ACT_HIGH

Returns

: void

Definition at line 169 of file GPIO_FW.c.

```
169                                                                 {
170     static uint8_t q = 0;   //Quantity of bounces
171     uint8_t j = 0;          //It captures changes
172
173     if (GPIO_GetPIN(port, pin, state))   // The key is pushed?
174         j = 0x01;               //Something is happening, the key is been pushed
175
```

```
176     if (buff_In ^ j) {          // If the key is pushed while q != BOUNCE
177         q++;                        // I change the buffer
178         if (q == BOUNCE) {
179             q = 0;
180             buff_In ^= 0x01;
181         }
182     } else
183         q = 0;
184 }
```

2.10.2.4 GPIO_DebounceUserKEY()

```
:void GPIO_DebounceUserKEY (
              void  )
```

: Firmware debounce for user key in board

: Use in SysTick_Handler or in some timer interrupt

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in]

Returns

: void

Definition at line 141 of file GPIO_FW.c.

```
141                                 {
142     static uint8_t q = 0;   //Quantity of bounces
143     uint8_t j = 0;          //It captures changes
144 
145     if (GPIO_GetPIN(UserKEY, ACT_LOW))    // The key is pushed?
146         j = 0x01;               //Something is happening, the key is been pushed
147 
148     if (buff_UserKEY ^ j) {         // If the key is pushed while q != BOUNCE
149         q++;                        // I change the buffer
150         if (q == BOUNCE) {
151             q = 0;
152             buff_UserKEY ^= 0x01;
153         }
154     } else
155         q = 0;
156 }
```

2.10.2.5 GPIO_Disable()

```
:void GPIO_Disable (
             void  )
```

: Disable GPIO0 and GPIO1

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 32 of file GPIO_FW.c.

```
32                          {
33     SYSAHBCLKCTRL0&= (~(1«6));
34     SYSAHBCLKCTRL0 &= (~(1«20));
35 }
```

2.10.2.6 GPIO_Enable()

```
:void GPIO_Enable (
             void  )
```

: Enable GPIO0 and GPIO1

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 19 of file GPIO_FW.c.

```
19                          {
20     SYSAHBCLKCTRL0|= (1«6);
21     SYSAHBCLKCTRL0 |= (1«20);
22 }
```

2.10.2.7 GPIO_GetPIN()

```
:uint8_t GPIO_GetPIN (
              uint8_t port,
              uint8_t pin,
              uint8_t dir )
```

: Return GPIO's input state

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31
	[in] uint8_t STATE : ACT_LOW,ACT_HIGH

Returns

: uint8_t : 1 pin == [state] , 0 pin != [state]

Definition at line 81 of file GPIO_FW.c.

```
81                                                          {
82     port = port * 32 + pin;
83     if ( GPIO_PBYTE[port] == state)
84         return 1;
85     else
86         return 0;
87 }
```

2.10.2.8 GPIO_SetDIR()

```
:void GPIO_SetDIR (
              uint8_t port,
              uint8_t pin,
              uint8_t dir )
```

: Choose GPIO as Input/Output

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31
	[in] uint8_t dir : INPUT,OUTPUT

Returns

: void

Definition at line 48 of file GPIO_FW.c.

```
48                                                      {
49     GPIO_DIRP[port] &= (~(1 « pin));
50     GPIO_DIRP[port] |= (dir « pin);
51 }
```

2.10.2.9 GPIO_SetModeCLKDIV()

```
:void GPIO_SetModeCLKDIV (
              uint8_t port,
              uint8_t pin,
              uint8_t mode )
```

: Select peripheral clock divider for input filter sampling clock

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port: PORT0,PORT1 : [in] uint8_t pin: 0,31 : [in] uint8_t mode: IOCONCLKDIV0 to IOCONCLKDIV6

Returns

: void

Definition at line 338 of file GPIO_FW.c.

```
338                                                                 {
339     uint8_t offset;
340     offset = GetOFFSET(port, pin);
341     IOCON_[offset] &= (~(0x07 « 13));
342     IOCON_[offset] |= (mode « 13);
343 }
```

2.10.2.10 GPIO_SetModeDAC()

```
:void GPIO_SetModeDAC (
             uint8_t port,
             uint8_t pin,
             uint8_t mode )
```

: Selects DAC mode

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port: PORT0,PORT1 : [in] uint8_t pin: 0,31 : [in] uint8_t mode: DAC_EN,DAC_DIS

Returns

: void

Definition at line 356 of file GPIO_FW.c.

```
356                                                             {
357     uint8_t offset;
358     offset = GetOFFSET(port, pin);
359     IOCON_[offset] &= (~(0x01 « 16));
360     IOCON_[offset] |= (mode « 16);
361 }
```

2.10.2.11 GPIO_SetModeFILTER()

```
:void GPIO_SetModeFILTER (
              uint8_t port,
              uint8_t pin,
              uint8_t mode )
```

: Digital filter sample mode

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port: PORT0,PORT1 : [in] uint8_t pin: 0,31 : [in] uint8_t mode: BYPASS_FILTER,CLK1_FILTER,CLK2_FILTER,CLK3_FILTER

Returns

: void

Definition at line 320 of file GPIO_FW.c.

```
320                                                                 {
321     uint8_t offset;
322     offset = GetOFFSET(port, pin);
323     IOCON_[offset] &= (~(0x03 « 11));
324     IOCON_[offset] |= (mode « 11);
325 }
```

2.10.2.12 GPIO_SetModeHYS()

```
:void GPIO_SetModeHYS (
              uint8_t port,
              uint8_t pin,
              uint8_t mode )
```

: Hysteresis

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port: PORT0,PORT1 : [in] uint8_t pin: 0,31 : [in] uint8_t mode:HYS_EN,HYS_DIS

Returns

: void

Definition at line 266 of file GPIO_FW.c.

```
266                                                                 {
267     uint8_t offset;
268     offset = GetOFFSET(port, pin);
269     IOCON_[offset] &= (~(0x01 « 5));
270     IOCON_[offset] |= (mode « 5);
271 }
```

2.10.2.13 GPIO_SetModeI2C()

```
:void GPIO_SetModeI2C (
             uint8_t port,
             uint8_t pin,
             uint8_t mode )
```

: Selects I2C mode

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port: PORT0,PORT1 : [in] uint8_t pin: 0,31 : [in] uint8_t mode:STD_MODE,STD_GPIO,FAST_MODE

Returns

: void

Definition at line 374 of file GPIO_FW.c.

```
374                                                                 {
375     uint8_t offset;
376     offset = GetOFFSET(port, pin);
377     IOCON_[offset] &= (~(0x03 « 8));
378     IOCON_[offset] |= (mode « 8);
379 }
```

2.10.2.14 GPIO_SetModeINPUT()

```
:void GPIO_SetModeINPUT (
              uint8_t port,
              uint8_t pin,
              uint8_t mode )
```

: on-chip pull-up/pull-down resistor

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port: PORT0,PORT1 : [in] uint8_t pin: 0,31 : [in] uint8_t mode:NO_PULL_UP_DOWN,PULL_DOWN,PULL_UP,REPEATER

Returns

: void

Definition at line 248 of file GPIO_FW.c.

```
248                                                                 {
249     uint8_t offset;
250     offset = GetOFFSET(port, pin);
251     IOCON_[offset] &= (~(0x03 << 3));
252     IOCON_[offset] |= (mode << 3);
253 }
```

2.10.2.15 GPIO_SetModeINV()

```
:void GPIO_SetModeINV (
              uint8_t port,
              uint8_t pin,
              uint8_t mode )
```

: Invert input

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port: PORT0,PORT1 : [in] uint8_t pin: 0,31 : [in] uint8_t mode: INV_INPUT,NOT_INV_INPUT

Returns

: void

Definition at line 284 of file GPIO_FW.c.

```
284                                                                 {
285     uint8_t offset;
286     offset = GetOFFSET(port, pin);
287     IOCON_[offset] &= (~(0x01 « 6));
288     IOCON_[offset] |= (mode « 6);
289 }
```

2.10.2.16 GPIO_SetModeOD()

```
:void GPIO_SetModeOD (
             uint8_t port,
             uint8_t pin,
             uint8_t mode )
```

: Open drain

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port: PORT0,PORT1 : [in] uint8_t pin: 0,31 : [in] uint8_t mode: OD_EN,OD_DIS

Returns

: void

Definition at line 302 of file GPIO_FW.c.

```
302                                                                 {
303     uint8_t offset;
304     offset = GetOFFSET(port, pin);
305     IOCON_[offset] &= (~(0x01 « 10));
306     IOCON_[offset] |= (mode « 10);
307 }
```

2.10.2.17 GPIO_SetOUT()

```
:void GPIO_SetOUT (
             uint8_t port,
             uint8_t pin )
```

: Put GPIO's out to 1

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31

Returns

: void

Definition at line 99 of file GPIO_FW.c.

```
99                                          {
100     GPIO_SETP[port] |= (1 « pin);
101 }
```

2.10.2.18 GPIO_SetPIN()

```
:void GPIO_SetPIN (
             uint8_t port,
             uint8_t pin,
             uint8_t dir )
```

: Choose GPIO's output state

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31
	[in] uint8_t state : LOW,HIGH

Returns

: void

Definition at line 64 of file GPIO_FW.c.

```
64                                                      {
65     port = port * 32 + pin;
66     GPIO_PBYTE[port] &= (~1);
67     GPIO_PBYTE[port] |= state;
68 }
```

2.10.2.19 GPIO_ToogleOUT()

```
:void GPIO_ToogleOUT (
             uint8_t port,
             uint8_t pin )
```

: Invert GPIO's out

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31

Returns

: void

Definition at line 127 of file GPIO_FW.c.

```
127                                             {
128     GPIO_NOTP[port] |= (1 « pin);
129 }
```

2.10.2.20 IOCONDisable()

```
:void IOCONDisable (
             void  )
```

: Disable IOCON

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in]

Returns

: void

Definition at line 208 of file GPIO_FW.c.

```
208                          {
209     SYSAHBCLKCTRL0&= (~(1<<18));
210 }
```

2.10.2.21 IOCONEnable()

```
:void IOCONEnable (
             void  )
```

: Enable IOCON

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in]

Returns

: void

Definition at line 195 of file GPIO_FW.c.

```
195                         {
196     SYSAHBCLKCTRL0|= (1«18);
197 }
```

2.10.3 Variable Documentation

2.10.3.1 offset

```
uint8_t offset[]
```

Initial value:

```
= { 0x044, 0x02C, 0x018, 0x014, 0x010, 0x00C, 0x040, 0x03C,
        0x038, 0x034, 0x020, 0x01C, 0x008, 0x004, 0x048, 0x028, 0x024, 0x000,
        0x078, 0x074, 0x070, 0x06C, 0x068, 0x064, 0x060, 0x05C, 0x058, 0x054,
        0x050, 0x0C8, 0x0CC, 0x08C, 0x090, 0x094, 0x098, 0x0A4, 0x0A8, 0x0AC,
        0x0B8, 0x0C4, 0x07C, 0x080, 0x0DC, 0x0D8, 0x084, 0x088, 0x09C, 0x0A0,
        0x0B0, 0x0B4, 0x0BC, 0x0C0, 0x0D0, 0x0D4 }
```

Definition at line 214 of file GPIO_FW.c.

2.11 source/GPIO_SW.c File Reference

: Software functions for GPIO

```
#include "Aplication.h"
```

Functions

- uint8_t GetUserKEY (void)

  : State of the user key in board
- uint8_t GetInput (void)

  : State of the input

Variables

- uint8_t **buff_UserKEY**
- uint8_t **buff_In**

2.11.1 Detailed Description

: Software functions for GPIO

: These functions avoid bouncing. Both must be used w/ GPIO_DebounceUserKEY or GPIO_Debounce

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

2.11.2 Function Documentation

2.11.2.1 GetInput()

```
:uint8_t GetInput (
             void  )
```

: State of the input

: Is necessary using GPIO_Debounce

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: uint8_t 1 if input pressed, 0 if input pressed

Definition at line 48 of file GPIO_SW.c.

```
48                      {
49     static uint8_t buff_before = 0x00;
50
51     if ( buff_In == 0x01 && buff_before == 0x00 ){
52         buff_before = 0x01;
53         return (1);
54     }
```

```
55     else if ( buff_In == 0x01 && buff_before == 0x01 )
56         return (0);
57     else if ( buff_In == 0x00 && buff_before == 0x01 )
58         return (0);
59     else
60         return (0);
61 }
```

2.11.2.2 GetUserKEY()

```
:uint8_t GetUserKEY (
              void  )
```

: State of the user key in board

: Is necessary using GPIO_DebounceUserKEY

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: uint8_t 1 if user key pressed, 0 if user key not

Definition at line 21 of file GPIO_SW.c.

```
21                          {
22     static uint8_t buff_before = 0x00;
23
24     if ( buff_UserKEY == 0x01 && buff_before == 0x00 ){
25         buff_before = 0x01;
26         return (1);
27     }
28     else if ( buff_UserKEY == 0x01 && buff_before == 0x01 )
29         return (0);
30     else if ( buff_UserKEY == 0x00 && buff_before == 0x01 ){
31         buff_before = 0x00;
32         return (0);
33     }
34     else
35         return (0);
36 }
```

2.12 source/mtb.c File Reference

MTB initialization file.

```
#include <cr_mtb_buffer.h>
```

Macros

- #define **__MTB_BUFFER_SIZE** 128

Functions

- **__CR_MTB_BUFFER** (__MTB_BUFFER_SIZE)

2.12.1 Detailed Description

MTB initialization file.

Symbols controlling behavior of this code... __MTB_DISABLE If this symbol is defined, then the buffer array for the MTB will not be created.

__MTB_BUFFER_SIZE Symbol specifying the sizer of the buffer array for the MTB. This must be a power of 2 in size, and fit into the available RAM. The MTB buffer will also be aligned to its 'size' boundary and be placed at the start of a RAM bank (which should ensure minimal or zero padding due to alignment).

__MTB_RAM_BANK Allows MTB Buffer to be placed into specific RAM bank. When this is not defined, the "default" (first if there are several) RAM bank is used.

2.13 source/SwitchMatrix_FW.c File Reference

: Firmware functions for SWM

```
#include "Aplication.h"
```

Functions

- void SWM (uint8_t port, uint8_t pin, uint8_t assign, uint8_t byte)
  : Assign movable functions for pin
- void SWM_PinEnable (uint8_t port, uint8_t pin, uint8_t ena)
  : Enable pin works as value passed in ena
- void SWM_Enable (void)
  : Enable SWM
- void SWM_Disable (void)
  : Disable SWM

2.13.1 Detailed Description

: Firmware functions for SWM

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

2.13.2 Function Documentation

2.13.2.1 SWM()

```
:void SWM (
             uint8_t port,
             uint8_t pin,
             uint8_t assign,
             uint8_t byte )
```

: Assign movable functions for pin

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31
	[in] uint8_t assign :
	[in] uint8_t byte : BYTE0,BYTE1,BYTE2,BYTE3

Returns

: void

Definition at line 22 of file SwitchMatrix_FW.c.

```
22                                                                  {
23     pin = pin + 0x20 * port; //PIO0[0:31] 0x00 to 0x1F PIO1[0:21] 0x1F to 0x35
24     PINASSIGN[assign] |= (pin « byte);
25 }
```

2.13.2.2 SWM_Disable()

```
:void SWM_Disable (
             void  )
```

: Disable SWM

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 67 of file SwitchMatrix_FW.c.

```
67                          {
68     SYSAHBCLKCTRL0&= (~(1«7));
69 }
```

2.13.2.3 SWM_Enable()

```
:void SWM_Enable (
             void  )
```

: Enable SWM

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 54 of file SwitchMatrix_FW.c.

```
54                          {
55     SYSAHBCLKCTRL0|= (1«7);
56 }
```

2.13.2.4 SWM_PinEnable()

```
:void SWM_PinEnable (
              uint8_t port,
              uint8_t pin,
              uint8_t ena )
```

: Enable pin works as value passed in ena

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint8_t port : PORT0,PORT1
	[in] uint8_t pin : 0,31
	[in] uint8_t ena : READ Page 143 UserManual. There are multiple choices

Returns

: void

Definition at line 38 of file SwitchMatrix_FW.c.

```
38                                                          {
39     if (port == PORT1)       //PIENABLE[0] -> PIO0_0 .... PIO1_2
40         if (pin < 3)         //PIENABLE10] -> PIO1_3 .... PIO1_21
41             port = PORT0;
42     PINENABLE[port] |= (1 « ena);
43 }
```

2.14 source/SYSCON_FW.c File Reference

: Firmware functions for SYSCON

```
#include "Aplication.h"
```

Functions

- void BoardClockRUN (void)

  : Runs clock at 30MHz
- void **ClockSetFroOscFREQ** (uint32_t freq)
- void **PowerDisablePD** (uint8_t en)
- void **CLOCK_SetFroOutClkSrc** (uint32_t src)

- void **CLOCK_Select** (uint8_t sel)
- void **CLOCK_InitSystemPll** (uint32_t freq, uint8_t src)
- uint32_t **CLOCK_GetSystemPLLInClockRate** (void)
- uint32_t **CLOCK_GetFroFreq** (void)
- uint32_t **FindSyestemPllPsel** (uint32_t outFreq)
- void **CLOCK_SetMainClkSrc** (uint32_t src)
- void **CLOCK_SetCoreSysClkDiv** (uint32_t value)

2.14.1 Detailed Description

: Firmware functions for SYSCON

: Only starts the board at 30MHz

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

2.14.2 Function Documentation

2.14.2.1 BoardClockRUN()

```
:void BoardClockRUN (
             void  )
```

: Runs clock at 30MHz

: Select clock from fro

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 19 of file SYSCON_FW.c.

```
19                          {
20     PowerDisablePD(FRO_OUT_PowerDown);
21     PowerDisablePD(FRO_PD);
22     ClockSetFroOscFREQ(F30MHz);
23     CLOCK_SetFroOutClkSrc(kCLOCK_FroSrcFroOsc);
24     PowerDisablePD(kPDRUNCFG_PD_SYSOSC);
25     CLOCK_Select(CLK_FROM_SYS_OSC);
26     CLOCK_InitSystemPll(FREQ30MHz, CLK_SYS_PLLSRCFRODIV);
27     CLOCK_SetMainClkSrc(kCLOCK_MainClkSrcFro);
28     CLOCK_SetCoreSysClkDiv(1U);
29 }
```

2.15 source/SysTick_FW.c File Reference

: Firmware functions for SysTick

```
#include "Aplication.h"
```

Functions

- void SysTick_Init (void)
  : Initialize the systick
- void SysTick_Off (void)
  : Stops the systick
- void SysTick_Set (uint32_t freq)
  *: Set the counter as freq*10mS -1*
- void SysTick_Handler (void)
  : Interrupt each 10mS

Variables

- uint32_t tick
  Declared in main.

2.15.1 Detailed Description

: Firmware functions for SysTick

: Only develop for 30MHz

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

2.15.2 Function Documentation

2.15.2.1 SysTick_Handler()

```
:void SysTick_Handler (
             void  )
```

: Interrupt each 10mS

: when the tick is out i know that happend time = tick∗10mS

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 61 of file SysTick_FW.c.

```
62 {
63
64     if (tick != 0U)
65         tick--;
66
67
68 }
```

2.15.2.2 SysTick_Init()

```
:void SysTick_Init (
             void  )
```

: Initialize the systick

: Enable SysTick, enable interrupt and set the counter

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 19 of file SysTick_FW.c.

```
19                      {
20     SysTick_Set(FRE30MHz);
21     SYST_CSR = SYSTICK_ENABLE_INTERRUPT_CLK;
22     SYST_CVR = 0;
23 }
```

2.15.2.3 SysTick_Off()

```
:  SysTick_Off (
             void  )
```

: Stops the systick

: disable SysTick, disable interrupt

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 34 of file SysTick_FW.c.

```
34                      {
35     SYST_CSR = SYSTICK_DISABLE;
36 }
```

2.15.2.4 SysTick_Set()

```
:void SysTick_Set (
             uint32_t freq )
```

: Set the counter as freq∗10mS -1

: Always use at 30MHz

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] uint32_t freq: FRE30MHz

Returns

: void

Definition at line 47 of file SysTick_FW.c.

```
47                                  {
48      SYST_RVR = freq*10 - 1; // 30MHz*10mS-1
49 }
```

2.15.3 Variable Documentation

2.15.3.1 tick

```
uint32_t tick  [extern]
```

Declared in main.

Declared in main.

Definition at line 20 of file 03-Semaphore.c.

Index