PWM

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 adc_chan Struct Reference

Public Attributes

- union {

      __RW uint32_t **_CHAN_THRSEL**

      struct {

        __RW uint32_t _CH0_THRSEL:1

        Compare against THR.

        __RW uint32_t _CH1_THRSEL:1

        Compare against THR.

        __RW uint32_t _CH2_THRSEL:1

        Compare against THR.

        __RW uint32_t _CH3_THRSEL:1

        Compare against THR.

        __RW uint32_t _CH4_THRSEL:1

        Compare against THR.

        __RW uint32_t _CH5_THRSEL:1

        Compare against THR.

        __RW uint32_t _CH6_THRSEL:1

        Compare against THR.

        __RW uint32_t _CH7_THRSEL:1

        Compare against THR.

        __RW uint32_t _CH8_THRSEL:1

        Compare against THR.

        __RW uint32_t _CH9_THRSEL:1

        Compare against THR.

        __RW uint32_t _CH10_THRSEL:1

        Compare against THR.

        __RW uint32_t _CH11_THRSEL:1

        Compare against THR.

        __RW uint32_t **RESERVED**:20

      }

  };

3.1.1 Detailed Description

Definition at line 145 of file ADC_FW.h.

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

- inc/ADC_FW.h

3.2 adc_ctrl_t Struct Reference

Public Attributes

- union {
    __RW uint32_t _CTRL
        < *Union between _CTRL and bit field; They're overlapped*
    struct {
        __RW uint32_t _CLKDIV:8
            Clock divided by this + 1 to produce sampling clock <= 30MHz.
        __RW uint32_t _ASYNCMODE:1
            Asyncronous operation mode.
        __RW uint32_t **RESERVED_0**:1
        __RW uint32_t _LPWRMODE:1
            Power down ADC while is not used.
        __RW uint32_t **RESERVED_1**:19
        __RW uint32_t _CALMODE:1
            Self calibration.
        __RW uint32_t **RESERVED_2**:1
    }
};

    < *Struct for handling adc configuration*

3.2.1 Detailed Description

Definition at line 56 of file ADC_FW.h.

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

- inc/ADC_FW.h

3.3 adc_seqX_ctrl Struct Reference

Public Attributes

- union {
    __RW uint32_t **_SEQx_CTRL**
    struct {
       __RW uint32_t _CHANNELS:12
          Select which channel will be sampled.
       __RW uint32_t _TRIGGER:3
          Select which HW trigger will start convertion.
       __RW uint32_t **RESERVED_0**:3
       __RW uint32_t _TRIGPOL:1
          Polarity of the input trigger.
       __RW uint32_t _SYNCBYPASS:1
          Byspass syncronization FF, so is slower.
       __RW uint32_t **_TSAMP**:5
       __RW uint32_t **RESERVED_1**:1
       __RW uint32_t _START:1
          Launch one pass.
       __RW uint32_t _BURST:1
          Sequence continuosly converted.
       __RW uint32_t _SINGLESTEP:1
          When start in 1 this converts only the next channel.
       __RW uint32_t _LOWPRIO:1
          Set priority for sequence A.
       __RW uint32_t _MODE:1
          Read global data or individual channel.
       __RW uint32_t _SEQx_ENA:1
          Enable sequence.
    }
  };

3.3.1 Detailed Description

Definition at line 74 of file ADC_FW.h.

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

- inc/ADC_FW.h

3.4 adc_seqX_gdat Struct Reference

Public Attributes

-

```
union {
    __RW uint32_t _SEQx_GDAT
    struct {
        __RW uint32_t _RESERVED_0:4
        __RW uint32_t _RESULT:12
            12 bit A/D convertion
        __RW uint32_t _THCMPRANGE:2
            Compare the result with thrn_low and thrn_high.
        __RW uint32_t _THCMPCROSS:2
            Indicates a crossing of the threshold.
        __RW uint32_t _RESERVED_1:6
        __RW uint32_t _CHN:4
            Indicates the channel converted.
        __RW uint32_t _OVERRUN:1
            If a new convertion was loaded and the previous was not read.
        __RW uint32_t _DATAVALID:1
            There's a new result.
    }
};
```

3.4.1   Detailed Description

Definition at line 111 of file ADC_FW.h.

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

- inc/ADC_FW.h

3.5   adc_thr Struct Reference

Public Attributes

-
```
union {
    __RW uint32_t _THRn_LH
    struct {
        __RW uint32_t _RESERVED_0:4
        __RW uint32_t _THR:12
            12bits for compare
        __RW uint32_t _RESERVED_1:16
    }
};
```

3.5.1   Detailed Description

Definition at line 132 of file ADC_FW.h.

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

- inc/ADC_FW.h

3.6 dac_cntval_t Struct Reference

Public Attributes

- union {
  - __RW uint32_t _CNTVAL
    - *< Union between _CR and bit field; They're overlapped*
  - struct {
    - __RW uint32_t _VALUE:16
      - *16b reload value for DAC interrupt/DMA timer*
    - __RW uint32_t **_RESERVED_0**:16
  - }
- };

  < Struct for handling adc configuration

3.6.1 Detailed Description

Definition at line 45 of file DAC_FW.h.

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

- inc/DAC_FW.h

3.7 dac_cr_t Struct Reference

Public Attributes

- union {
  - __RW uint32_t _CR
    - *< Union between _CR and bit field; They're overlapped*
  - struct {
    - __RW uint32_t **_RESERVED_0**:6
    - __RW uint32_t _VALUE:10
      - *3.3V∗Value/1024 = VDAC*
    - __RW uint32_t _BIAS:1
      - *Settling time.*
    - __RW uint32_t **_LPWRMODE**:15
  - }
- };

  < Struct for handling adc configuration

3.7.1 Detailed Description

Definition at line 15 of file DAC_FW.h.

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

- inc/DAC_FW.h

3.8 dac_ctrl_t Struct Reference

Public Attributes

-
    union {
        __RW uint32_t _CTRL
        $<$ *Union between _CR and bit field; They're overlapped*
        struct {
            __RW uint32_t _INT_DMA_REQ:1
            DMA interrupt request.
            __RW uint32_t _DBLBUF_ENA:1
            Double buffer.
            __RW uint32_t _CNT_ENA:1
            Time-out counter operation.
            __RW uint32_t _DMA_ENA:1
            DMA acces.
            __RW uint32_t _**RESERVED_0**:28
        }
    };

    $<$ *Struct for handling adc configuration*

3.8.1 Detailed Description

Definition at line 30 of file DAC_FW.h.

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

- inc/DAC_FW.h

Chapter 4

File Documentation

4.1 inc/ADC_FW.h File Reference

: Firmware functions ADC

Classes

- struct adc_ctrl_t
- struct adc_seqX_ctrl
- struct adc_seqX_gdat
- struct adc_thr
- struct adc_chan

Macros

- #define **MASK_ADC_SYSCON** 4
- #define **ADC_SYSAHB** 24
- #define ADC_0 PORT0,7,14

    POT1 on board; 14 is the bit in PINENABLE.
- #define **ADC_ADD** ((__RW uint32_t ∗) 0x4001C000UL)
- #define **_ADC_CTRL** ADC_ADD[0]
- #define **_ADC_SEQA_CTRL** ADC_ADD[2]
- #define **_ADC_SEQB_CTRL** ADC_ADD[3]
- #define **_ADC_SEQA_GDAT** ADC_ADD[4]
- #define **_ADC_SEQB_GDAT** ADC_ADD[5]
- #define **_ADC_DAT0** ADC_ADD[8]
- #define **_ADC_DAT1** ADC_ADD[9]
- #define **_ADC_DAT2** ADC_ADD[10]
- #define **_ADC_DAT3** ADC_ADD[11]
- #define **_ADC_DAT4** ADC_ADD[12]
- #define **_ADC_DAT5** ADC_ADD[13]
- #define **_ADC_DAT6** ADC_ADD[14]
- #define **_ADC_DAT7** ADC_ADD[15]
- #define **_ADC_DAT8** ADC_ADD[16]
- #define **_ADC_DAT9** ADC_ADD[17]
- #define **_ADC_DAT10** ADC_ADD[18]

- #define **_ADC_DAT11** ADC_ADD[19]
- #define **_ADC_THR0_LOW** ADC_ADD[20]
- #define **_ADC_THR1_LOW** ADC_ADD[21]
- #define **_ADC_THR0_HIGH_** ADC_ADD[22]
- #define **_ADC_THR1_HIGH_** ADC_ADD[23]
- #define **_ADC_CHAN_THRSEL** ADC_ADD[24]
- #define **ADC_INTEN** ADC_ADD[25]
- #define **ADC_FLAGS** ADC_ADD[26]
- #define **ADC_TRM** ADC_ADD[27]
- #define ADC_CTRL ((__RW adc_ctrl_t ∗) 0x4001C000UL)

  Pointer to a struct in that memory.
- #define ADC_SEQA_CTRL ((__RW adc_seqX_ctrl ∗) 0x4001C008UL)

  Pointer to a struct in that memory.
- #define ADC_SEQB_CTRL ((__RW adc_seqX_ctrl ∗) 0x4001C00CUL)

  Pointer to a struct in that memory.
- #define ADC_SEQA_GDAT ((__RW adc_seqX_gdat ∗) 0x4001C010UL)

  Pointer to a struct in that memory.
- #define ADC_SEQB_GDAT ((__RW adc_seqX_gdat ∗) 0x4001C014UL)

  Pointer to a struct in that memory.
- #define ADC_DAT0 ((__RW adc_seqX_gdat ∗) 0x4001C020UL)

  Pointer to a struct in that memory.
- #define ADC_DAT1 ((__RW adc_seqX_gdat ∗) 0x4001C024UL)

  Pointer to a struct in that memory.
- #define ADC_DAT2 ((__RW adc_seqX_gdat ∗) 0x4001C028UL)

  Pointer to a struct in that memory.
- #define ADC_DAT3 ((__RW adc_seqX_gdat ∗) 0x4001C02CUL)

  Pointer to a struct in that memory.
- #define ADC_DAT4 ((__RW adc_seqX_gdat ∗) 0x4001C030UL)

  Pointer to a struct in that memory.
- #define ADC_DAT5 ((__RW adc_seqX_gdat ∗) 0x4001C034UL)

  Pointer to a struct in that memory.
- #define ADC_DAT6 ((__RW adc_seqX_gdat ∗) 0x4001C038UL)

  Pointer to a struct in that memory.
- #define ADC_DAT7 ((__RW adc_seqX_gdat ∗) 0x4001C03CUL)

  Pointer to a struct in that memory.
- #define ADC_DAT8 ((__RW adc_seqX_gdat ∗) 0x4001C040UL)

  Pointer to a struct in that memory.
- #define ADC_DAT9 ((__RW adc_seqX_gdat ∗) 0x4001C044UL)

  Pointer to a struct in that memory.
- #define ADC_DAT10 ((__RW adc_seqX_gdat ∗) 0x4001C048UL)

  Pointer to a struct in that memory.
- #define ADC_DAT11 ((__RW adc_seqX_gdat ∗) 0x4001C04CUL)

  Pointer to a struct in that memory.
- #define ADC_THR0_LOW ((__RW adc_thr ∗) 0x4001C050UL)

  Pointer to a struct in that memory.
- #define ADC_THR1_LOW ((__RW adc_thr ∗) 0x4001C054UL)

  Pointer to a struct in that memory.
- #define ADC_THR0_HIGH ((__RW adc_thr ∗) 0x4001C058UL)

  Pointer to a struct in that memory.
- #define ADC_THR1_HIGH ((__RW adc_thr ∗) 0x4001C05CUL)

  Pointer to a struct in that memory.

- #define ADC_CHAN_THRSEL ((__RW adc_chan *) 0x4001C060UL)

    Pointer to a struct in that memory.
- #define MASK_SEQA_INTEN 1<<0

    Interrupt after each conv.
- #define MASK_SEQB_INTEN 1<<1

    Interrupt after each conv.
- #define MASK_ISE_ADC_SEQA 1<<16

    Enable Interrupt NVIC.
- #define MASK_ISE_ADC_SEQB 1<<17

    Enable Interrupt NVIC.

Functions

- void ADC_Init (uint8_t port, uint8_t pin, uint8_t ena)

    : Initialize ADC on a pin
- void ADC_Power (void)

    : Power ADC
- void ADC_Enable (void)

    : Enable clock in ADC
- void ADC_Disable (void)

    : Disable clock in ADC

4.1.1 Detailed Description

: Firmware functions ADC

: 12 bits convertion

Author

    : Tobias Bavasso Piizzi

Date

    : 08/01/2021

4.1.2 Function Documentation

4.1.2.1 ADC_Disable()

```
void ADC_Disable (
            void  )
```

: Disable clock in ADC

:

Author

    : Tobias Bavasso Piizzi

Date

    : 08/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 90 of file ADC_FW.c.

```
90                          {
91      SYSAHBCLKCTRL0&= (~(1«ADC_SYSAHB));
92 }
```

4.1.2.2 ADC_Enable()

```
void ADC_Enable (
            void  )
```

: Enable clock in ADC

:

Author

: Tobias Bavasso Piizzi

Date

: 08/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 77 of file ADC_FW.c.

```
77                             {
78      SYSAHBCLKCTRL0|= (1«ADC_SYSAHB);
79 }
```

4.1.2.3 ADC_Init()

```
void ADC_Init (
            uint8_t port,
```

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

: Initialize ADC on a pin

: Continuos conversion of POTE in board

Author

: Tobias Bavasso Piizzi

Date

: 08/01/2021

Parameters

[in] uint8_t port: PORT0,PORT1
[in] uint8_t pin: 0,31
[in] uint8_t en: bit to enable in PINENABLE (page 143 UM)

Returns

: void

$<$ Enable CLOCK in SYSAHB

$<$ Enable service interrupt

$<$ Interrupt after conversion finish

$<$ Enable Switch Matrix

$<$ Enable pin in SWN as AnalogInput

$<$ Disable Switch Matrix

$<$ Power in SYSCON

$<$ Div = 0

$<$ Sync

$<$ OFF

$<$ OFF

$<$ Sample CH0

$<$ No hardware trigger

$<$ Positive trigger

$<$ Enable sync

< Individual end of conversion

< Start,enable set on the same line first time

Definition at line 23 of file ADC_FW.c.

```
23                                                                      {
24
25      ADC_Enable();
26      ISER0|= MASK_ISE_ADC_SEQA;
27      ADC_INTEN|= MASK_SEQA_INTEN;
28      SWM_Enable();
29      SWM_PinEnable(port, pin, ena);
30      SWM_Disable();
31      ADC_Power();
32
33
34
35      ADC_CTRL->_CLKDIV = 0x00;
36      ADC_CTRL->_ASYNCMODE = 0;
37      ADC_CTRL->_LPWRMODE = 0;
38      ADC_CTRL->_CALMODE = 0;
39
40      ADC_SEQA_CTRL->_CHANNELS      = 0x01;
41      ADC_SEQA_CTRL->_TRIGGER       = 0x00;
42      ADC_SEQA_CTRL->_TRIGPOL       = 0x1;
43      ADC_SEQA_CTRL->_SYNCBYPASS        = 0x0;
44      ADC_SEQA_CTRL->_TSAMP         = 0x00;
45      ADC_SEQA_CTRL->_START         = 0;
46      ADC_SEQA_CTRL->_BURST         = 0;
47      ADC_SEQA_CTRL->_SINGLESTEP        = 0x0;
48      ADC_SEQA_CTRL->_LOWPRIO       = 0x0;
49      ADC_SEQA_CTRL->_MODE          = 0;
50      ADC_SEQA_CTRL->_SEQx_ENA      = 0;
51      _ADC_SEQA_CTRL |= ((0b100001) « 26);
52 }
```

4.1.2.4 ADC_Power()

```
void ADC_Power (
            void  )
```

: Power ADC

:

Author

: Tobias Bavasso Piizzi

Date

: 08/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 63 of file ADC_FW.c.

```
63                        {
64      PDRUNCFG&= (~(1 « MASK_ADC_SYSCON));
65
66 }
```

4.2 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"
#include "Disp7Seg_FW.h"
#include "Disp7Seg_SW.h"
#include "ADC_FW.h"
#include "DAC_FW.h"
#include "DAC_SW.h"
#include "PWM_FW.h"
```

Functions

- void LPC_Init (void)

    : Initialize the board
- void GPIO_Init (void)

    : Initialize the GPIO

4.2.1 Detailed Description

: Functions used in main

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

4.2.2 Function Documentation

4.2.2.1 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     GPIO_SetDIR(UserKEY, INPUT);
37     GPIO_SetDIR(LedGREEN, OUTPUT);
38     GPIO_SetDIR(LedBLUE, OUTPUT);
39
40     GPIO_SetPIN(LedGREEN, LED_OFF);
41     GPIO_SetPIN(LedBLUE, LED_OFF);
42 }
```

4.2.2.2 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  }
```

4.3 inc/DAC_FW.h File Reference

: Firmware functions for DAC 10 bits

Classes

- struct dac_cr_t
- struct dac_ctrl_t
- struct dac_cntval_t

Macros

- #define DAC0_CR ((__RW dac_cr_t ∗) 0x40014000UL)

  Pointer to a struct in that memory.
- #define DAC1_CR ((__RW dac_cr_t ∗) 0x40018000UL)

  Pointer to a struct in that memory.
- #define DAC0_CTRL ((__RW dac_ctrl_t ∗) 0x40014004UL)

  Pointer to a struct in that memory.
- #define DAC1_CTRL ((__RW dac_ctrl_t ∗) 0x40018004UL)

  Pointer to a struct in that memory.
- #define DAC0_CNTVAL ((__RW dac_cntval_t ∗) 0x40014008UL)

  Pointer to a struct in that memory.
- #define DAC1_CNTVAL ((__RW dac_cntval_t ∗) 0x40018008UL)

  Pointer to a struct in that memory.
- #define DACOUT0 PORT0,17,26

  26 is the bit in PINENABLE
- #define **MASK_DAC0_SYSCON** 13
- #define **MASK_DAC1_SYSCON** 14
- #define **DAC0_SYSAHB** 27
- #define **DAC1_SYSAHB** 27
- #define **DACMODE_IOCON** 1<<17
- #define DAC_ENABLE_IOCON 1

  Enable DACOUT in IOCON.

Functions

- void **DAC0_Init** (uint8_t port, uint8_t pin, uint8_t ena)
- void DAC_Power (void)
    - *: Initialize DAC0*
- void DAC_Enable (void)
    - *: Enable clock in DAC0,DAC1*
- void DAC_Disable (void)
    - *: Disable clock in DAC0,DAC1*
- void **DAC0_SetValue** (void)

4.3.1 Detailed Description

: Firmware functions for DAC 10 bits

:

Author

    : Tobias Bavasso Piizzi

Date

    : 12/01/2021

4.3.2 Function Documentation

4.3.2.1 DAC_Disable()

```
void DAC_Disable (
            void  )
```

: Disable clock in DAC0,DAC1

:

Author

    : Tobias Bavasso Piizzi

Date

    : 12/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 97 of file DAC_FW.c.

```
97                         {
98      SYSAHBCLKCTRL0 &= (~(1 « DAC0_SYSAHB));
99      SYSAHBCLKCTRL1 &= (~(1 « DAC1_SYSAHB));
100 }
```

4.3.2.2 DAC_Enable()

```
void DAC_Enable (
            void  )
```

: Enable clock in DAC0,DAC1

:

Author

: Tobias Bavasso Piizzi

Date

: 12/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 83 of file DAC_FW.c.

```
83                         {
84      SYSAHBCLKCTRL0 |= (1 « DAC0_SYSAHB);
85      SYSAHBCLKCTRL1 |= (1 « DAC1_SYSAHB);
86 }
```

4.3.2.3 DAC_Power()

```
void DAC_Power (
            void  )
```

: Initialize DAC0

: Power DAC0,DAC1

:

Author

: Tobias Bavasso Piizzi

Date

: 12/01/2021

Parameters

	[in] uint8_t port: PORT0,PORT1
	[in] uint8_t pin: 0,31
	[in] uint8_t en: bit to enable in PINENABLE (page 143 UM)

Returns

: void

:

Author

: Tobias Bavasso Piizzi

Date

: 12/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 69 of file DAC_FW.c.

```
69                  {
70    PDRUNCFG &= (~(1 « MASK_DAC0_SYSCON));
71    PDRUNCFG &= (~(1 « MASK_DAC1_SYSCON));
72 }
```

4.4 inc/DAC_SW.h File Reference

: Software functions for DAC 10 bits

Functions

- void SetDAC0 (uint16_t val)

    : Select the voltage output

4.4.1 Detailed Description

: Software functions for DAC 10 bits

:

Author

: Tobias Bavasso Piizzi

Date

: 12/01/2021

4.4.2 Function Documentation

4.4.2.1 SetDAC0()

```
void SetDAC0 (
            uint16_t val )
```

: Select the voltage output

: Vout = (3.3V ∗ val)/1024

Author

: Tobias Bavasso Piizzi

Date

: 12/01/2021

Parameters

		[in] uint16_t val: 10 bits

Returns

: void

Definition at line 19 of file DAC_SW.c.
```
19                                    {
20    buffDac0 = (val & 0x3FF);   // To be sure buffDac0 is 10 bits
21 }
```

4.5 inc/Disp7Seg_FW.h File Reference

: Firmware functions for DISP7SEG

Macros

- #define SEG_A PORT0,21

   Pin to connect segA.
- #define SEG_B PORT0,22

   Pin to connect segB.
- #define SEG_C PORT0,16

   Pin to connect segC.
- #define SEG_D PORT0,17

   Pin to connect segD.
- #define SEG_E PORT0,18

   Pin to connect segE.
- #define SEG_F PORT0,20

   Pin to connect segF.
- #define SEG_G PORT0,19

   Pin to connect segG.
- #define SEG_DP PORT0,23

   Pin to connect segDP.
- #define TR_D1 PORT0,0

   Pin to connect transistor DISP1.
- #define TR_D0 PORT0,1

   Pin to connect transistor DISP0.
- #define DIGITS 2

   Number of displays.
- #define **DIGIT_0** 0
- #define **DIGIT_1** 1

Functions

- void DISP7SEG_Init (void)

   : Set pins for display as out
- void DISP_Sweep (void)

   : Refresh the display 7Seg (2 Disp)

4.5.1 Detailed Description

: Firmware functions for DISP7SEG

:

Author

   : Tobias Bavasso Piizzi

Date

   : 07/01/2021

4.5.2 Function Documentation

4.5.2.1 DISP7SEG_Init()

```
void DISP7SEG_Init (
            void  )
```

: Set pins for display as out

:

Author

: Tobias Bavasso Piizzi

Date

: 07/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 19 of file Disp7Seg_FW.c.

```
19          {
20      GPIO_SetDIR(SEG_A,  OUTPUT);
21      GPIO_SetDIR(SEG_B,  OUTPUT);
22      GPIO_SetDIR(SEG_C,  OUTPUT);
23      GPIO_SetDIR(SEG_D,  OUTPUT);
24      GPIO_SetDIR(SEG_E,  OUTPUT);
25      GPIO_SetDIR(SEG_F,  OUTPUT);
26      GPIO_SetDIR(SEG_G,  OUTPUT);
27      GPIO_SetDIR(TR_D0,  OUTPUT);
28      GPIO_SetDIR(TR_D1,  OUTPUT);
29
30      GPIO_ClearOUT(SEG_A);
31      GPIO_ClearOUT(SEG_B);
32      GPIO_ClearOUT(SEG_C);
33      GPIO_ClearOUT(SEG_D);
34      GPIO_ClearOUT(SEG_E);
35      GPIO_ClearOUT(SEG_F);
36      GPIO_ClearOUT(SEG_G);
37      GPIO_ClearOUT(TR_D0);
38      GPIO_ClearOUT(TR_D1);
39 }
```

4.5.2.2 DISP_Sweep()

```
void DISP_Sweep (
            void  )
```

: Refresh the display 7Seg (2 Disp)

: Is necessary to be used in SysTick_Handler

Author

: Tobias Bavasso Piizzi

Date

: 07/01/2021

Parameters

	[in] void

Returns

: void

< Number of disp

< Turn off transistor

< Turn off transistor

< Next time sweep other disp

< Reset the digits

Definition at line 51 of file Disp7Seg_FW.c.

```
51                         {
52     uint8_t aux;
53     static uint8_t digit = 0;
54
55     GPIO_ClearOUT(TR_D0);
56     GPIO_ClearOUT(TR_D1);
57
58     aux = buff_Disp7[digit];
59
60     GPIO_SetPIN( SEG_A, ((aux >> 0) & (uint8_t) 0x01));
61     GPIO_SetPIN( SEG_B, ((aux >> 1) & (uint8_t) 0x01));
62     GPIO_SetPIN( SEG_C, ((aux >> 2) & (uint8_t) 0x01));
63     GPIO_SetPIN( SEG_D, ((aux >> 3) & (uint8_t) 0x01));
64     GPIO_SetPIN( SEG_E, ((aux >> 4) & (uint8_t) 0x01));
65     GPIO_SetPIN( SEG_F, ((aux >> 5) & (uint8_t) 0x01));
66     GPIO_SetPIN( SEG_G, ((aux >> 6) & (uint8_t) 0x01));
67     GPIO_SetPIN( SEG_DP, ((aux >> 7) & (uint8_t) 0x01));
68
69     switch (digit) {
70     case DIGIT_0:
71         GPIO_SetOUT(TR_D0);
72         break;
73     case DIGIT_1:
74         GPIO_SetOUT(TR_D1);
75         break;
76     default:
77         digit = 0;
78         GPIO_SetOUT(TR_D0);
79         break;
80     }
81
82     digit++;
83     digit %= DIGITS;
84
85 }
```

4.6 inc/Disp7Seg_SW.h File Reference

: Software functions for DISP7SEG

Functions

- void Display (uint8_t val)

    : Writes on Disp7Seg

4.6.1 Detailed Description

: Software functions for DISP7SEG

:

Author

: Tobias Bavasso Piizzi

Date

: 07/01/2021

4.6.2 Function Documentation

4.6.2.1 Display()

```
void Display (
            uint8_t val )
```

: Writes on Disp7Seg

: High lever of layers

Author

: Tobias Bavasso Piizzi

Date

: 07/01/2021

Parameters

	[in] uint8_t val: 0 to 99

Returns

: void

$<$ Disable SysTick INT

$<$ Enable SysTick INT

Definition at line 38 of file Disp7Seg_SW.c.
```
38                              {
39      uint8_t i;
40      uint8_t auxDisp[DIGITS];
41
42      for (i = 0; i < DIGITS; i++) {
43          auxDisp[i] = Digits_to_BCD7seg[val % 10];
44          val /= 10;
45      }
46      for (i = 0; i < DIGITS; i++) {
47          SYSTICK_INT_DIS;
48          buff_Disp7[i] = auxDisp[i];
49          SYSTICK_INT_EN;
50      }
51
52 }
```

4.7 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*

4.7.1 Detailed Description

: Firmware functions for GPIO

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

4.7.2 Function Documentation

4.7.2.1 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 }
```

4.7.2.2 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 }
```

4.7.2.3 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 }
```

4.7.2.4 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 }
```

4.7.2.5 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 }
```

4.7.2.6 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 }
```

4.7.2.7 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 }
```

4.7.2.8 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 }
```

4.7.2.9 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 }
```

4.7.2.10 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 }
```

4.7.2.11 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 }
```

4.7.2.12 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 }
```

4.7.2.13 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 }
```

4.7.2.14 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 }
```

4.7.2.15 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 }
```

4.7.2.16 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 }
```

4.7.2.17 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 }
```

4.7.2.18 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 }
```

4.7.2.19 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 }
```

4.7.2.20 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 }
```

4.7.2.21 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 }
```

4.8 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*

4.8.1 Detailed Description

: Software functions for GPIO

: These are functions in a higher layer of abstraction

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

4.8.2 Function Documentation

4.8.2.1 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 }
```

4.8.2.2 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 }
```

4.9 inc/LPC845.h File Reference

: Declarations for type of data

Macros

- #define __**R** volatile const
- #define __**W** volatile
- #define __**RW** volatile
- #define _**ISER** ((__RW uint32_t *) 0xE000E100UL)
- #define **ISER0** _ISER[0]

Typedefs

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

4.9.1 Detailed Description

: Declarations for type of data

: Only contains macros

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

4.10 inc/PWM_FW.h File Reference

: Firmware functions PWM

Macros

- #define SCT_ADD ((__RW uint32_t ∗) 0x50004000UL)

    Base address.
- #define SCT_CONFIG SCT_ADD[0]

    OFFSET 0x000.
- #define SCT_CTRL SCT_ADD[1]

    OFFSET 0x004.
- #define **SCT_CTRL_L** ((__RW uint16_t ∗) 0x50004004UL)[0]
- #define **SCT_CTRL_H** ((__RW uint16_t ∗) 0x50004006UL)
- #define SCT_LIMIT SCT_ADD[2]

    OFFSET 0x008.
- #define **SCT_LIMIT_L** ((__RW uint16_t ∗) 0x50004008UL)
- #define **SCT_LIMIT_H** ((__RW uint16_t ∗) 0x5000400AUL)
- #define SCT_HALT SCT_ADD[3]

    OFFSET 0x00C.
- #define **SCT_HALT_L** ((__RW uint16_t ∗) 0x5000400CUL)
- #define **SCT_HALT_H** ((__RW uint16_t ∗) 0x5000400EUL)
- #define SCT_STOP SCT_ADD[4]

    OFFSET 0x010.
- #define **SCT_STOP_L** ((__RW uint16_t ∗) 0x50004010UL)
- #define **SCT_STOP_H** ((__RW uint16_t ∗) 0x50004012UL)
- #define SCT_START SCT_ADD[5]

    OFFSET 0x014.
- #define SCT_COUNT SCT_ADD[16]

    OFFSET 0x040.
- #define SCT_STATE SCT_ADD[17]

    OFFSET 0x044.
- #define SCT_INPUT SCT_ADD[18]

    OFFSET 0x048.
- #define SCT_REGMODE SCT_ADD[19]

    OFFSET 0x04C.
- #define SCT_OUTPUT SCT_ADD[20]

    OFFSET 0x050.
- #define SCT_OUTPUTDIRCTRL SCT_ADD[21]

    OFFSET 0x054.
- #define SCT_RES SCT_ADD[22]

    OFFSET 0x058.
- #define SCT_DMAREQ0 SCT_ADD[23]

    OFFSET 0x05C.
- #define SCT_DMAREQ1 SCT_ADD[24]

    OFFSET 0x060.
- #define SCT_EVEN SCT_ADD[60]

    OFFSET 0x0F0.
- #define SCT_EVFLAG SCT_ADD[61]

    OFFSET 0x0F4.
- #define SCT_CONEN SCT_ADD[62]

    OFFSET 0x0F8.
- #define SCT_CONFLAG SCT_ADD[63]

    OFFSET 0x0FC.
- #define SCT_MATCH0 SCT_ADD[64]

OFFSET 0x100.

- #define **SCT_MATCH0_L** ((__RW uint16_t ∗) 0x50004100UL)[0]
- #define SCT_MATCH1 SCT_ADD[65]

  OFFSET 0x104.
- #define SCT_MATCH2 SCT_ADD[66]

  OFFSET 0x108.
- #define SCT_MATCH3 SCT_ADD[67]

  OFFSET 0x10C.
- #define SCT_MATCH4 SCT_ADD[68]

  OFFSET 0x110.
- #define SCT_MATCH5 SCT_ADD[69]

  OFFSET 0x114.
- #define SCT_MATCH6 SCT_ADD[70]

  OFFSET 0x118.
- #define SCT_MATCH7 SCT_ADD[71]

  OFFSET 0x11C.
- #define SCT_CAP0 SCT_ADD[64]

  OFFSET 0x100.
- #define SCT_CAP1 SCT_ADD[65]

  OFFSET 0x104.
- #define SCT_CAP2 SCT_ADD[66]

  OFFSET 0x108.
- #define SCT_CAP3 SCT_ADD[67]

  OFFSET 0x10C.
- #define SCT_CAP4 SCT_ADD[68]

  OFFSET 0x110.
- #define SCT_CAP5 SCT_ADD[69]

  OFFSET 0x114.
- #define SCT_CAP6 SCT_ADD[70]

  OFFSET 0x118.
- #define SCT_CAP7 SCT_ADD[71]

  OFFSET 0x11C.
- #define SCT_MATCHREL0 SCT_ADD[128]

  OFFSET 0x200.
- #define SCT_MATCHREL1 SCT_ADD[129]

  OFFSET 0x204.
- #define SCT_MATCHREL2 SCT_ADD[130]

  OFFSET 0x208.
- #define SCT_MATCHREL3 SCT_ADD[131]

  OFFSET 0x20C.
- #define SCT_MATCHREL4 SCT_ADD[132]

  OFFSET 0x210.
- #define SCT_MATCHREL5 SCT_ADD[133]

  OFFSET 0x214.
- #define SCT_MATCHREL6 SCT_ADD[134]

  OFFSET 0x218.
- #define SCT_MATCHREL7 SCT_ADD[135]

  OFFSET 0x21C.
- #define SCT_EV0_STATE SCT_ADD[192]

  OFFSET 0x300.
- #define SCT_EV0_CTRL SCT_ADD[193]

OFFSET 0x304.

- #define SCT_EV1_STATE SCT_ADD[194]

  OFFSET 0x308.
- #define SCT_EV1_CTRL SCT_ADD[195]

  OFFSET 0x30C.
- #define SCT_EV2_STATE SCT_ADD[196]

  OFFSET 0x310.
- #define SCT_EV2_CTRL SCT_ADD[197]

  OFFSET 0x314.
- #define SCT_EV3_STATE SCT_ADD[198]

  OFFSET 0x318.
- #define SCT_EV3_CTRL SCT_ADD[199]

  OFFSET 0x31C.
- #define SCT_EV4_STATE SCT_ADD[200]

  OFFSET 0x320.
- #define SCT_EV4_CTRL SCT_ADD[201]

  OFFSET 0x324.
- #define SCT_EV5_STATE SCT_ADD[202]

  OFFSET 0x328.
- #define SCT_EV5_CTRL SCT_ADD[203]

  OFFSET 0x32C.
- #define SCT_EV6_STATE SCT_ADD[204]

  OFFSET 0x330.
- #define SCT_EV6_CTRL SCT_ADD[205]

  OFFSET 0x334.
- #define SCT_EV7_STATE SCT_ADD[206]

  OFFSET 0x338.
- #define SCT_EV7_CTRL SCT_ADD[207]

  OFFSET 0x33C.
- #define SCT_OUT0SET SCT_ADD[320]

  OFFSET 0x500.
- #define SCT_OUT0CLR SCT_ADD[321]

  OFFSET 0x504.
- #define SCT_OUT1SET SCT_ADD[322]

  OFFSET 0x508.
- #define SCT_OUT1CLR SCT_ADD[323]

  OFFSET 0x50C.
- #define SCT_OUT2SET SCT_ADD[324]

  OFFSET 0x510.
- #define SCT_OUT2CLR SCT_ADD[325]

  OFFSET 0x514.
- #define SCT_OUT3SET SCT_ADD[326]

  OFFSET 0x518.
- #define SCT_OUT3CLR SCT_ADD[327]

  OFFSET 0x51C.
- #define SCT_OUT4SET SCT_ADD[328]

  OFFSET 0x520.
- #define SCT_OUT4CLR SCT_ADD[329]

  OFFSET 0x524.
- #define SCT_OUT5SET SCT_ADD[330]

  OFFSET 0x528.

- #define SCT_OUT5CLR SCT_ADD[331]

    OFFSET 0x52C.
- #define SCT_OUT6SET SCT_ADD[332]

    OFFSET 0x530.
- #define SCT_OUT6CLR SCT_ADD[333]

    OFFSET 0x534.
- #define **SCT_SYSAHB** 8
- #define **SCT_PRESETCTRL** 8
- #define **PWM** PORT0,0,SCT_OUT0,BYTE3
- #define **OUT0** 0
- #define **PWM_FREQ** 10
- #define **PWM_PERIOD** (1000000/(PWM_FREQ∗2))
- #define **PWM_STEP** (PWM_PERIOD/100)

Functions

- void **SCT_Init** (uint8_t port, uint8_t pin, uint8_t assign, uint8_t byte)
- void PWM_Set (uint8_t val)

    : Set duty on PWM
- void SCT_Preset (void)

    : Clear SCT reset control
- void SCT_Enable (void)

    : Enable clock in PWM
- void SCT_Disable (void)

    : Disable clock in PWM

4.10.1 Detailed Description

: Firmware functions PWM

: 12 bits convertion

Author

    : Tobias Bavasso Piizzi

Date

    : 17/01/2021

4.10.2 Function Documentation

4.10.2.1 PWM_Set()

```
void PWM_Set (
            uint8_t val )
```

: Set duty on PWM

:

Author

    : Tobias Bavasso Piizzi

Date

    : 25/01/2021

Parameters

	[in] uint8_t val : 0 100

Returns

: void

Definition at line 66 of file PWM_FW.c.

```
66                                {
67     if(val == 0)
68         SCT_MATCHREL1 = 0;
69     else if(val < 100)
70         SCT_MATCHREL1 = ((PWM_STEP*val) - 1UL);
71     else
72         SCT_MATCHREL1 = PWM_PERIOD - 2;
73 }
```

4.10.2.2 SCT_Disable()

```
void SCT_Disable (
            void  )
```

: Disable clock in PWM

:

Author

: Tobias Bavasso Piizzi

Date

: 17/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 113 of file PWM_FW.c.

```
113                              {
114     SYSAHBCLKCTRL0&= (~(1«SCT_SYSAHB));
115 }
```

4.10.2.3 SCT_Enable()

```
void SCT_Enable (
            void  )
```

: Enable clock in PWM

:

Author

: Tobias Bavasso Piizzi

Date

: 17/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 100 of file PWM_FW.c.

```
100                         {
101     SYSAHBCLKCTRL0|= (1«SCT_SYSAHB);
102 }
```

4.10.2.4 SCT_Preset()

```
void SCT_Preset (
            void  )
```

: Clear SCT reset control

:

Author

: Tobias Bavasso Piizzi

Date

: 17/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 86 of file PWM_FW.c.

```
86                    {
87     PRESETCTRL0 &= (~(1«SCT_PRESETCTRL));
88     PRESETCTRL0 |= (1«SCT_PRESETCTRL);
89 }
```

4.11 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

4.11.1 Detailed Description

: Firmware functions for SWM

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

4.11.2 Enumeration Type Documentation

4.11.2.1 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 };
```

4.11.2.2 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 };
```

4.11.2.3 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 };
```

4.11.2.4 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 };
```

4.11.3 Function Documentation

4.11.3.1 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] &= (~(0xff « (byte*8)));
25      PINASSIGN[assign] |= (pin « (byte*8));
26 }
```

4.11.3.2 SWM_Disable()

```
void SWM_Disable (
            void  )
```

: Disable SWM

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 68 of file SwitchMatrix_FW.c.

```
68                          {
69      SYSAHBCLKCTRL0&= (~(1«7));
70 }
```

4.11.3.3 SWM_Enable()

```
void SWM_Enable (
            void  )
```

: Enable SWM

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 55 of file SwitchMatrix_FW.c.

```
55              {
56    SYSAHBCLKCTRL0|= (1«7);
57 }
```

4.11.3.4 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 39 of file SwitchMatrix_FW.c.

```
39                                              {
40     if (port == PORT1)      //PIENABLE[0] -> PIO0_0 .... PIO1_2
41        if (pin < 3)         //PIENABLE10] -> PIO1_3 .... PIO1_21
42            port = PORT0;
43     PINENABLE[port] &= (~(1 « ena));
44 }
```

4.12 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)

4.12.1 Detailed Description

: Firmware functions for SYSCON

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

4.12.2 Function Documentation

4.12.2.1 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 }
```

4.13   inc/SysTick_FW.h File Reference

: Firmware functions for SysTick

Macros

- #define TICK_OUT_1S 100-1
  - *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 **SYSTICK_INT_DIS** SYST_CSR &= ∼0x02;
- #define **SYSTICK_INT_EN** SYST_CSR = SYSTICK_ENABLE_INTERRUPT_CLK;
- #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

4.13.1 Detailed Description

: Firmware functions for SysTick

: Used for 30 MHz

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

4.13.2 Function Documentation

4.13.2.1 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 }
```

4.13.2.2 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 }
```

4.13.2.3 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 }
```

4.14 source/07-PWM.c File Reference

: Example of PWM use

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

Functions

- int main (void)

  : Main Function

Variables

- uint32_t tick = TICK_OUT_1S

  Var for systick.
- uint32_t conv = 0

  Var for ADC.
- uint8_t duty = 0

  Declared in main.

4.14.1 Detailed Description

: Example of PWM use

:

Author

: Tobias Bavasso Piizzi

Date

: 13/01/2021

4.14.2 Function Documentation

4.14.2.1 main()

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

: Main Function

: initialize the system and stay in the while

Author

: Tobias Bavasso Piizzi

Date

: 13/01/2021

Parameters

	[in] void

Returns

: int

Definition at line 24 of file 07-PWM.c.

```
24          {
25      LPC_Init();
26      SCT_Init(PWM);
27      PWM_Set(duty);
28
29      while(1) {
30
31          if(tick == 0){
32              tick = TICK_OUT_1S;
33              duty++;
34              PWM_Set(duty);
35              if( duty == 100)
36                  duty = 0;
37          }
38
39      }
40      return 0 ;
41 }
```

4.14.3 Variable Documentation

4.14.3.1 tick

```
uint32_t tick = TICK_OUT_1S
```

Var for systick.

Declared in main.

Definition at line 19 of file 07-PWM.c.

4.15 source/ADC_FW.c File Reference

: Firmware functions ADC

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

Functions

- void ADC_Init (uint8_t port, uint8_t pin, uint8_t ena)
  - *: Initialize ADC on a pin*
- void ADC_Power (void)
  - *: Power ADC*
- void ADC_Enable (void)
  - *: Enable clock in ADC*
- void ADC_Disable (void)
  - *: Disable clock in ADC*
- void ADC0_SEQA_IRQHandler (void)
  - *: Interruption for ADC*

Variables

- uint32_t tick
  - *Var for systick.*
- uint32_t conv
  - *Var for ADC.*

4.15.1 Detailed Description

: Firmware functions ADC

: 12 bits convertion

Author

: Tobias Bavasso Piizzi

Date

: 08/01/2021

4.15.2 Function Documentation

4.15.2.1 ADC0_SEQA_IRQHandler()

```
:void ADC0_SEQA_IRQHandler (
            void  )
```

: Interruption for ADC

: Interrupt when some channel finishes its conversion

Author

: Tobias Bavasso Piizzi

Date

: 10/01/2021

Parameters

	[in] void

Returns

: void

$<$ Clean flags

$<$ Read global data

$<$ Make an average

$<$ Start a new conversion

Definition at line 104 of file ADC_FW.c.

```
104                                              {
105      static uint8_t i    = 0;
106      static uint32_t sum = 0;
107
108      (void) _ADC_SEQA_GDAT;
109
110      sum += ADC_SEQA_GDAT->_RESULT;
111      i++;
112      if( i == 0xFF ){
113          conv    = sum/i ;
114          i       = 0;
115          sum     = 0;
116      }
117      ADC_SEQA_CTRL->_START  = 1;
118 }
```

4.15.2.2 ADC_Disable()

```
:void ADC_Disable (
                void  )
```

: Disable clock in ADC

:

Author

> : Tobias Bavasso Piizzi

Date

> : 08/01/2021

Parameters

	[in] void

Returns

> : void

Definition at line 90 of file ADC_FW.c.

```
90                    {
91    SYSAHBCLKCTRL0&= (~(1«ADC_SYSAHB));
92 }
```

4.15.2.3 ADC_Enable()

```
:void ADC_Enable (
                void  )
```

: Enable clock in ADC

:

Author

> : Tobias Bavasso Piizzi

Date

> : 08/01/2021

Parameters

	[in] void

Returns

    : void

Definition at line 77 of file ADC_FW.c.

```
77                                    {
78     SYSAHBCLKCTRL0|= (1«ADC_SYSAHB);
79 }
```

4.15.2.4 ADC_Init()

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

: Initialize ADC on a pin

: Continuos conversion of POTE in board

Author

    : Tobias Bavasso Piizzi

Date

    : 08/01/2021

Parameters

[in] uint8_t port: PORT0,PORT1	
[in] uint8_t pin: 0,31	
[in] uint8_t en: bit to enable in PINENABLE (page 143 UM)	

Returns

    : void

$<$ Enable CLOCK in SYSAHB

$<$ Enable service interrupt

$<$ Interrupt after conversion finish

$<$ Enable Switch Matrix

$<$ Enable pin in SWN as AnalogInput

$<$ Disable Switch Matrix

$<$ Power in SYSCON

$<$ Div = 0

$<$ Sync

$<$ OFF

$<$ OFF

$<$ Sample CH0

$<$ No hardware trigger

$<$ Positive trigger

$<$ Enable sync

$<$ Individual end of conversion

$<$ Start,enable set on the same line first time

Definition at line 23 of file ADC_FW.c.

```
23                                                        {
24
25       ADC_Enable();
26       ISER0|= MASK_ISE_ADC_SEQA;
27       ADC_INTEN|= MASK_SEQA_INTEN;
28       SWM_Enable();
29       SWM_PinEnable(port, pin, ena);
30       SWM_Disable();
31       ADC_Power();
32
33
34
35       ADC_CTRL->_CLKDIV = 0x00;
36       ADC_CTRL->_ASYNCMODE = 0;
37       ADC_CTRL->_LPWRMODE = 0;
38       ADC_CTRL->_CALMODE = 0;
39
40       ADC_SEQA_CTRL->_CHANNELS      = 0x01;
41       ADC_SEQA_CTRL->_TRIGGER       = 0x00;
42       ADC_SEQA_CTRL->_TRIGPOL       = 0x1;
43       ADC_SEQA_CTRL->_SYNCBYPASS       = 0x0;
44       ADC_SEQA_CTRL->_TSAMP         = 0x00;
45       ADC_SEQA_CTRL->_START         = 0;
46       ADC_SEQA_CTRL->_BURST         = 0;
47       ADC_SEQA_CTRL->_SINGLESTEP       = 0x0;
48       ADC_SEQA_CTRL->_LOWPRIO       = 0x0;
49       ADC_SEQA_CTRL->_MODE          = 0;
50       ADC_SEQA_CTRL->_SEQx_ENA      = 0;
51       _ADC_SEQA_CTRL |= ((0b100001) « 26);
52 }
```

4.15.2.5 ADC_Power()

```
:void ADC_Power (
          void  )
```

: Power ADC

:

Author

: Tobias Bavasso Piizzi

Date

: 08/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 63 of file ADC_FW.c.

```
63                    {
64      PDRUNCFG&= (~(1 « MASK_ADC_SYSCON));
65
66 }
```

4.16 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

Variables

- uint32_t tick
    - Declared in main.

4.16.1 Detailed Description

: Functions used in main

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

4.16.2 Function Documentation

4.16.2.1 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     GPIO_SetDIR(UserKEY, INPUT);
37     GPIO_SetDIR(LedGREEN, OUTPUT);
38     GPIO_SetDIR(LedBLUE, OUTPUT);
39
40     GPIO_SetPIN(LedGREEN, LED_OFF);
41     GPIO_SetPIN(LedBLUE, LED_OFF);
42 }
```

4.16.2.2 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 }
```

4.16.3 Variable Documentation

4.16.3.1 tick

```
uint32_t tick  [extern]
```

Declared in main.

Declared in main.

Definition at line 19 of file 07-PWM.c.

4.17 source/DAC_FW.c File Reference

: Firmware functions for DAC

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

Functions

- void **DAC0_Init** (uint8_t port, uint8_t pin, uint8_t ena)
- void **DAC0_SetValue** (void)
- void DAC_Power (void)
    - *: Initialize DAC0*
- void DAC_Enable (void)
    - *: Enable clock in DAC0,DAC1*
- void DAC_Disable (void)
    - *: Disable clock in DAC0,DAC1*

Variables

- uint16_t **buffDac0**

4.17.1 Detailed Description

: Firmware functions for DAC

:

Author

    : Tobias Bavasso Piizzi

Date

    : 12/01/2021

4.17.2 Function Documentation

4.17.2.1 DAC_Disable()

```
:void DAC_Disable (
          void  )
```

: Disable clock in DAC0,DAC1

:

Author

    : Tobias Bavasso Piizzi

Date

    : 12/01/2021

Parameters

	[in] void

Returns

    : void

Definition at line 97 of file DAC_FW.c.

```
97                        {
98    SYSAHBCLKCTRL0 &= (~(1 « DAC0_SYSAHB));
99    SYSAHBCLKCTRL1 &= (~(1 « DAC1_SYSAHB));
100 }
```

4.17.2.2 DAC_Enable()

```
:void DAC_Enable (
            void  )
```

: Enable clock in DAC0,DAC1

:

Author

    : Tobias Bavasso Piizzi

Date

    : 12/01/2021

Parameters

	[in] void

Returns

    : void

Definition at line 83 of file DAC_FW.c.
```
83                        {
84    SYSAHBCLKCTRL0 |= (1 « DAC0_SYSAHB);
85    SYSAHBCLKCTRL1 |= (1 « DAC1_SYSAHB);
86 }
```

4.17.2.3 DAC_Power()

```
:void DAC_Power (
            void  )
```

: Initialize DAC0

: Power DAC0,DAC1

:

Author

    : Tobias Bavasso Piizzi

Date

    : 12/01/2021

Parameters

	[in] uint8_t port: PORT0,PORT1
	[in] uint8_t pin: 0,31
	[in] uint8_t en: bit to enable in PINENABLE (page 143 UM)

Returns

: void

:

Author

: Tobias Bavasso Piizzi

Date

: 12/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 69 of file DAC_FW.c.

```
69                      {
70      PDRUNCFG &= (~(1 « MASK_DAC0_SYSCON));
71      PDRUNCFG &= (~(1 « MASK_DAC1_SYSCON));
72 }
```

4.18   source/DAC_SW.c File Reference

: Software functions for DAC

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

Functions

• void SetDAC0 (uint16_t val)
    : Select the voltage output

Variables

• uint16_t **buffDac0**

4.18.1 Detailed Description

: Software functions for DAC

:

Author

: Tobias Bavasso Piizzi

Date

: 12/01/2021

4.18.2 Function Documentation

4.18.2.1 SetDAC0()

```
:void SetDAC0 (
            uint16_t val )
```

: Select the voltage output

: Vout = (3.3V ∗ val)/1024

Author

: Tobias Bavasso Piizzi

Date

: 12/01/2021

Parameters

	[in] uint16_t val: 10 bits

Returns

: void

Definition at line 19 of file DAC_SW.c.

```
19                               {
20     buffDac0 = (val & 0x3FF);   // To be sure buffDac0 is 10 bits
21 }
```

4.19 source/Disp7Seg_FW.c File Reference

: Firmware functions for DISP7SEG

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

Functions

- void DISP7SEG_Init (void)

  : Set pins for display as out
- void DISP_Sweep (void)

  : Refresh the display 7Seg (2 Disp)

Variables

- __RW uint8_t buff_Disp7 [ ]

  Display buffer.

4.19.1 Detailed Description

: Firmware functions for DISP7SEG

:

Author

: Tobias Bavasso Piizzi

Date

: 07/01/2021

4.19.2 Function Documentation

4.19.2.1 DISP7SEG_Init()

```
:void DISP7SEG_Init (
            void )
```

: Set pins for display as out

:

Author

: Tobias Bavasso Piizzi

Date

: 07/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 19 of file Disp7Seg_FW.c.

```
19                        {
20      GPIO_SetDIR(SEG_A, OUTPUT);
21      GPIO_SetDIR(SEG_B, OUTPUT);
22      GPIO_SetDIR(SEG_C, OUTPUT);
23      GPIO_SetDIR(SEG_D, OUTPUT);
24      GPIO_SetDIR(SEG_E, OUTPUT);
25      GPIO_SetDIR(SEG_F, OUTPUT);
26      GPIO_SetDIR(SEG_G, OUTPUT);
27      GPIO_SetDIR(TR_D0, OUTPUT);
28      GPIO_SetDIR(TR_D1, OUTPUT);
29
30      GPIO_ClearOUT(SEG_A);
31      GPIO_ClearOUT(SEG_B);
32      GPIO_ClearOUT(SEG_C);
33      GPIO_ClearOUT(SEG_D);
34      GPIO_ClearOUT(SEG_E);
35      GPIO_ClearOUT(SEG_F);
36      GPIO_ClearOUT(SEG_G);
37      GPIO_ClearOUT(TR_D0);
38      GPIO_ClearOUT(TR_D1);
39 }
```

4.19.2.2 DISP_Sweep()

```
:void DISP_Sweep (
            void  )
```

: Refresh the display 7Seg (2 Disp)

: Is necessary to be used in SysTick_Handler

Author

: Tobias Bavasso Piizzi

Date

: 07/01/2021

Parameters

	[in] void

Returns

: void

< Number of disp

< Turn off transistor

< Turn off transistor

< Next time sweep other disp

< Reset the digits

Definition at line 51 of file Disp7Seg_FW.c.

```
51                      {
52      uint8_t aux;
53      static uint8_t digit = 0;
54
55      GPIO_ClearOUT(TR_D0);
56      GPIO_ClearOUT(TR_D1);
57
58      aux = buff_Disp7[digit];
59
60      GPIO_SetPIN( SEG_A, ((aux » 0) & (uint8_t) 0x01));
61      GPIO_SetPIN( SEG_B, ((aux » 1) & (uint8_t) 0x01));
62      GPIO_SetPIN( SEG_C, ((aux » 2) & (uint8_t) 0x01));
63      GPIO_SetPIN( SEG_D, ((aux » 3) & (uint8_t) 0x01));
64      GPIO_SetPIN( SEG_E, ((aux » 4) & (uint8_t) 0x01));
65      GPIO_SetPIN( SEG_F, ((aux » 5) & (uint8_t) 0x01));
66      GPIO_SetPIN( SEG_G, ((aux » 6) & (uint8_t) 0x01));
67      GPIO_SetPIN( SEG_DP, ((aux » 7) & (uint8_t) 0x01));
68
69      switch (digit) {
70      case DIGIT_0:
71          GPIO_SetOUT(TR_D0);
72          break;
73      case DIGIT_1:
74          GPIO_SetOUT(TR_D1);
75          break;
76      default:
77          digit = 0;
78          GPIO_SetOUT(TR_D0);
79          break;
80      }
81
82      digit++;
83      digit %= DIGITS;
84
85 }
```

4.20   source/Disp7Seg_SW.c File Reference

: Software functions for DISP7SEG

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

Functions

- void Display (uint8_t val)
    - *: Writes on Disp7Seg*

Variables

- __RW uint8_t buff_Disp7 [DIGITS]
    - *Buffer de display.*
- uint8_t Digits_to_BCD7seg [ ]
- __RW uint8_t **tick_Disp7**

4.20.1 Detailed Description

: Software functions for DISP7SEG

:

Author

: Tobias Bavasso Piizzi

Date

: 07/01/2021

4.20.2 Function Documentation

4.20.2.1 Display()

```
:void Display (
            uint8_t val )
```

: Writes on Disp7Seg

: High lever of layers

Author

: Tobias Bavasso Piizzi

Date

: 07/01/2021

Parameters

	[in] uint8_t val: 0 to 99

Returns

: void

< Disable SysTick INT

< Enable SysTick INT

Definition at line 38 of file Disp7Seg_SW.c.
38   {

```
39    uint8_t i;
40    uint8_t auxDisp[DIGITS];
41
42    for (i = 0; i < DIGITS; i++) {
43        auxDisp[i] = Digits_to_BCD7seg[val % 10];
44        val /= 10;
45    }
46    for (i = 0; i < DIGITS; i++) {
47        SYSTICK_INT_DIS;
48        buff_Disp7[i] = auxDisp[i];
49        SYSTICK_INT_EN;
50    }
51
52 }
```

4.20.3 Variable Documentation

4.20.3.1 Digits_to_BCD7seg

```
uint8_t Digits_to_BCD7seg[]
```

Initial value:
```
= { 0x3f, 0x06, 0x5B, 0x4f, 0x66, 0x6D, 0x7C, 0x07,
        0x7f, 0x67 }
```

Tabla de conversion bcd a 7 segmentos Codigo bcd a b c d e f g dp 0 1 1 1 1 1 1 0 0 1 0 1 1 0 0 0 0 2 1 1 0 1 1 0 1 3 1 1 1 1 0 0 1 4 0 1 1 0 0 1 1 5 1 0 1 1 0 1 1 6 0 0 1 1 1 1 1 7 1 1 1 0 0 0 0 8 1 1 1 1 1 1 1 9 1 1 1 0 0 1 1

Definition at line 26 of file Disp7Seg_SW.c.

4.21 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** [ ]

4.21.1 Detailed Description

: Firmware functions for GPIO

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

4.21.2 Function Documentation

4.21.2.1 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 }
```

4.21.2.2 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 }
```

4.21.2.3 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 }
```

4.21.2.4 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 }
```

4.21.2.5 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 }
```

4.21.2.6 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 }
```

4.21.2.7 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 }
```

4.21.2.8 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 }
```

4.21.2.9 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 }
```

4.21.2.10 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 }
```

4.21.2.11 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 }
```

4.21.2.12 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  }
```

4.21.2.13 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  }
```

4.21.2.14 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 }
```

4.21.2.15 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 }
```

4.21.2.16 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 }
```

4.21.2.17 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 }
```

4.21.2.18 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 }
```

4.21.2.19 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 }
```

4.21.2.20 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 }
```

4.21.2.21 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 }
```

4.21.3 Variable Documentation

4.21.3.1 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.

4.22 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**

4.22.1 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

4.22.2 Function Documentation

4.22.2.1 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 }
```

4.22.2.2 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 }
```

4.23   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)

4.23.1 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.

4.24 source/PWM_FW.c File Reference

: Firmware functions PWM

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

Functions

- void **SCT_Init** (uint8_t port, uint8_t pin, uint8_t assign, uint8_t byte)
- void PWM_Set (uint8_t val)
    - *: Set duty on PWM*
- void SCT_Preset (void)
    - *: Clear SCT reset control*
- void SCT_Enable (void)
    - *: Enable clock in PWM*
- void SCT_Disable (void)
    - *: Disable clock in PWM*

4.24.1 Detailed Description

: Firmware functions PWM

: 12 bits convertion

Author

: Tobias Bavasso Piizzi

Date

: 17/01/2021

4.24.2 Function Documentation

4.24.2.1 PWM_Set()

```
:void PWM_Set (
            uint8_t val )
```

: Set duty on PWM

:

Author

: Tobias Bavasso Piizzi

Date

: 25/01/2021

Parameters

	[in] uint8_t val : 0 100

Returns

: void

Definition at line 66 of file PWM_FW.c.

```
66                         {
67     if(val == 0)
68         SCT_MATCHREL1 = 0;
69     else if(val < 100)
70         SCT_MATCHREL1 = ((PWM_STEP*val) - 1UL);
71     else
72         SCT_MATCHREL1 = PWM_PERIOD - 2;
73 }
```

4.24.2.2 SCT_Disable()

```
:void SCT_Disable (
            void )
```

: Disable clock in PWM

:

Author

: Tobias Bavasso Piizzi

Date

: 17/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 113 of file PWM_FW.c.
```
113                        {
114     SYSAHBCLKCTRL0&= (~(1«SCT_SYSAHB));
115 }
```

4.24.2.3 SCT_Enable()

```
:void SCT_Enable (
            void  )
```

: Enable clock in PWM

:

Author

: Tobias Bavasso Piizzi

Date

: 17/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 100 of file PWM_FW.c.
```
100                       {
101     SYSAHBCLKCTRL0|= (1«SCT_SYSAHB);
102 }
```

4.24.2.4 SCT_Preset()

```
:void SCT_Preset (
            void  )
```

: Clear SCT reset control

:

Author

: Tobias Bavasso Piizzi

Date

: 17/01/2021

Parameters

	[in] void

Returns

: void

Definition at line 86 of file PWM_FW.c.

```
86                        {
87     PRESETCTRL0 &= (~(1«SCT_PRESETCTRL));
88     PRESETCTRL0 |= (1«SCT_PRESETCTRL);
89 }
```

4.25 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

4.25.1 Detailed Description

: Firmware functions for SWM

:

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

4.25.2 Function Documentation

4.25.2.1 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] &= (~(0xff « (byte*8)));
25     PINASSIGN[assign] |= (pin « (byte*8));
26 }
```

4.25.2.2 SWM_Disable()

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

: Disable SWM

:

Author

    : Tobias Bavasso Piizzi

Date

    : 04/01/2021

Parameters

	[in] void

Returns

    : void

Definition at line 68 of file SwitchMatrix_FW.c.

```
68                    {
69     SYSAHBCLKCTRL0&= (~(1«7));
70 }
```

4.25.2.3 SWM_Enable()

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

: Enable SWM

:

Author

    : Tobias Bavasso Piizzi

Date

    : 04/01/2021

Parameters

	[in] void

Returns

    : void

Definition at line 55 of file SwitchMatrix_FW.c.

```
55                    {
56     SYSAHBCLKCTRL0|= (1«7);
57 }
```

4.25.2.4 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 39 of file SwitchMatrix_FW.c.
```
39                                                           {
40    if (port == PORT1)      //PIENABLE[0] -> PIO0_0 .... PIO1_2
41        if (pin < 3)        //PIENABLE10] -> PIO1_3 .... PIO1_21
42            port = PORT0;
43    PINENABLE[port] &= (~(1 « ena));
44 }
```

4.26 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)

4.26.1 Detailed Description

: Firmware functions for SYSCON

: Only starts the board at 30MHz

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

4.26.2 Function Documentation

4.26.2.1 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 }
```

4.27 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.
- uint8_t duty
    - Declared in main.

4.27.1 Detailed Description

: Firmware functions for SysTick

: Only develop for 30MHz

Author

: Tobias Bavasso Piizzi

Date

: 04/01/2021

4.27.2 Function Documentation

4.27.2.1 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 62 of file SysTick_FW.c.

```
62                                    {
63
64      if (tick > 0U){
65          tick--;
66      }
67
68
69
70 }
```

4.27.2.2 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 }
```

4.27.2.3 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 }
```

4.27.2.4 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 }
```

4.27.3 Variable Documentation

4.27.3.1 tick

```
uint32_t tick  [extern]
```

Declared in main.

Declared in main.

Definition at line 19 of file 07-PWM.c.

Index