

PWM task

task1

Generated by Doxygen 1.8.18

1 File Index	1
1.1 File List	1
2 File Documentation	3
2.1 GPIO.c File Reference	3
2.1.1 Function Documentation	3
2.1.1.1 Gpinit()	3
2.2 GPIO.h File Reference	3
2.2.1 Function Documentation	4
2.2.1.1 Gpinit()	4
2.3 main.c File Reference	4
2.3.1 Function Documentation	4
2.3.1.1 main()	4
2.4 Timer1.c File Reference	4
2.4.1 Function Documentation	5
2.4.1.1 Timer1init()	5
2.5 Timer1.h File Reference	6
2.5.1 Macro Definition Documentation	6
2.5.1.1 F_CPU	6
2.5.2 Function Documentation	6
2.5.2.1 Timer1init()	6
Index	9

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

GPIO.c	3
GPIO.h	3
main.c	4
Timer1.c	4
Timer1.h	6

Chapter 2

File Documentation

2.1 GPIO.c File Reference

```
#include "GPIO.h"
```

Functions

- void [Gpininit](#) ()
this function is responsible for setting the B1 and B2 pins as output pins

2.1.1 Function Documentation

2.1.1.1 Gpininit()

```
void Gpininit ( )
```

this function is responsible for setting the B1 and B2 pins as output pins

2.2 GPIO.h File Reference

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

Functions

- void [Gpininit](#) ()
this function is responsible for setting the B1 and B2 pins as output pins

2.2.1 Function Documentation

2.2.1.1 Gpininit()

```
void Gpininit ( )
```

this function is responsible for setting the B1 and B2 pins as output pins

2.3 main.c File Reference

```
#include <avr/io.h>
#include "GPIO.h"
#include "Timer1.h"
```

Functions

- int [main](#) (void)

2.3.1 Function Documentation

2.3.1.1 main()

```
int main (
    void )
```

2.4 Timer1.c File Reference

```
#include "Timer1.h"
```

Functions

- void [Timer1init](#) ()

this function is responsible for generating the 2-complementary signals with the injected dead time

2.4.1 Function Documentation

2.4.1.1 Timer1init()

```
void Timer1init ( )
```

this function is responsible for generating the 2-complementary signals with the injected dead time

internal frequency = 8MHZ;

desired frequency = 5KHZ;

$(2 * \text{prescaler} * \text{topValue}) = \text{internal} / \text{desired} = 8\text{MHZ} / 5\text{MHz} = 1600;$

$\text{prescaler} * \text{topValue} = 800;$

since, the topValue can be held by the ICR

then, we don't have to use a prescaler

$\text{ICR} = 800; \text{prescaler} = 1:1; \rightarrow$ setting the prescaler to 1:1 through the bits CS12:CS11:CS10

duty cycle $D = \text{onTime} / (\text{onTime} + \text{offTime});$

$\text{OCSR}_x = D * \text{topValue};$

for A, let the duty cycle be 50% i.e. $\text{onTime} = 100$ unit, $\text{offTime} = 200$ unit;

$\text{OCSRA} = 50\% * 800 = 400;$

lets think about how to configure B to have a dead time of 10 units:

having a dead time of 10 units means that the onTime of B would be less than that of A by $2*10$ which is 80 units

hence, the duty cycle of B is $80/200 = 40\%$

$\text{OCSRB} = 40\% * 800 = 320;$

we need to set our PWM mode to phase correct mode with a topValue of ICR1

WGM13 : WGM12 : WGM11 : WGM10

1 0 1 0

in order to obtain two complementary signals,

we should set one of the ports to the inverted mode and the other to the non-inverted mode

COM1x1 : COM1x0

1 0 \rightarrow non-inverted i.e. clear \rightarrow set

1 1 \rightarrow inverted i.e. set \rightarrow clear

2.5 Timer1.h File Reference

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

Macros

- `#define F_CPU 8000000ul`
setting the CPU frequency to 8MHz.

Functions

- `void Timer1init ()`
this function is responsible for generating the 2-complementary signals with the injected dead time

2.5.1 Macro Definition Documentation

2.5.1.1 F_CPU

```
#define F_CPU 8000000ul
```

setting the CPU frequency to 8MHz.

2.5.2 Function Documentation

2.5.2.1 Timer1init()

```
void Timer1init ( )
```

this function is responsible for generating the 2-complementary signals with the injected dead time

internal frequency = 8MHz;

desired frequency = 5KHz;

$(2 * \text{prescaler} * \text{topValue}) = \text{internal} / \text{desired} = 8\text{MHz} / 5\text{MHz} = 1600;$

$\text{prescaler} * \text{topValue} = 800;$

since, the topValue can be held by the ICR

then, we don't have to use a prescaler

ICR = 800; prescaler = 1:1; --> setting the prescaler to 1:1 through the bits CS12:CS11:CS10

duty cycle $D = \text{onTime} / (\text{onTime} + \text{offTime})$;

$\text{OCSR}_x = D * \text{topValue}$;

for A, let the duty cycle be 50% i.e. onTime = 100 unit, offTime = 200 unit;

$\text{OCSRA} = 50\% * 800 = 400$;

lets think about how to configure B to have a dead time of 10 units:

having a dead time of 10 units means that the onTime of B would be less than that of A by $2 * 10$ which is 80 units

hence, the duty cycle of B is $80/200 = 40\%$

$\text{OCSRB} = 40\% * 800 = 320$;

we need to set our PWM mode to phase correct mode with a topValue of ICR1

WGM13 : WGM12 : WGM11 : WGM10

1 0 1 0

in order to obtain two complementary signals,

we should set one of the ports to the inverted mode and the other to the non-inverted mode

COM1x1 : COM1x0

1 0 --> non-inverted i.e. clear --> set

1 1 --> inverted i.e. set --> clear

Index

F_CPU
 Timer1.h, [6](#)

Gpininit
 GPIO.c, [3](#)
 GPIO.h, [4](#)

GPIO.c, [3](#)
 Gpininit, [3](#)

GPIO.h, [3](#)
 Gpininit, [4](#)

main
 main.c, [4](#)
main.c, [4](#)
 main, [4](#)

Timer1.c, [4](#)
 Timer1init, [5](#)

Timer1.h, [6](#)
 F_CPU, [6](#)
 Timer1init, [6](#)

Timer1init
 Timer1.c, [5](#)
 Timer1.h, [6](#)