PWM task

task1

Generated by Doxygen 1.8.18

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

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 withe 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 withe the injected dead time

internal frequency = 8MHZ;

desired frequency = 5KHZ;

(2 ∗ prescaler ∗ topValue) = internal/ desired = 8MHZ/5MHz = 1600;

prescaler ∗ topValue = 800;

since, the topValue can be held by the ICR

then, we don't have to use a prescaler

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

duty cycle D = onTime / (onTime + offTime);

OCSRx = D ∗ topValue;

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

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%

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

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 withe 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 withe the injected dead time

internal frequency = 8MHZ;

desired frequency = 5KHZ;

(2 ∗ prescaler ∗ topValue) = internal/ desired = 8MHZ/5MHz = 1600;

prescaler ∗ topValue = 800;

since, the topValue can be held by the ICR

then, we don't have to use a prescaler

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

duty cycle D = onTime / (onTime + offTime);

OCSRx = D $*$ topValue;

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

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%

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