by-wire). The set-point will be transferred between the two motor controllers using UDP messages. The actual state of the controller and its history will be published as live graphs over the HTTP protocol.

MOTOR CONTROL

Generated by Doxygen 1.9.3

Chapter 1

motorControl

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

client.c
This file represents the client, i.e. the engine that must receive its position and constantly send data to the other engine (server) to move when necessary . . . . . ??

ircs.c
This file contain the functions that are common in the client.c and server.c in order to have more organizated all the project. The declaration of the functions is in the "ircs.h" file . . . . . ??

ircs.h
This file contain the declaration of the functions that are common in the client.c and server.c in order to have more organizated all the project . . . . . ??

server.c
This file represents the server, i.e. the motor that must receive the position of the motor of the client and move to the position received . . . . . ??

Chapter 3

File Documentation

3.1 client.c File Reference

This file represents the client, i.e. the engine that must receive its position and constantly send data to the other engine (server) to move when necessary.

```
#include <taskLib.h>
#include <stdio.h>
#include <kernelLib.h>
#include <semLib.h>
#include <intLib.h>
#include <iv.h>
#include <xlnx_zynq7k.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <inetLib.h>
#include <sockLib.h>
#include "ircs.h"
```

Functions

- void init_connection ()
- void send_data ()
- void get_position ()
- void motorClient (void)

Variables

- volatile int irc_a
- volatile int irc_b
- int sockd
- struct sockaddr_in my_addr srv_addr
- char buf [32]
- int motorPos

3.1.1 Detailed Description

This file represents the client, i.e. the engine that must receive its position and constantly send data to the other engine (server) to move when necessary.

Author

ROC BENAIGES MORAGREGA

3.1.2 Function Documentation

3.1.2.1 get_position()

```
void get_position ( )
```

This method will be used to constantly obtain the position of the engine to check if it reads it correctly.

Author

ROC BENAIGES MORAGREGA

3.1.2.2 init_connection()

```
void init_connection ( )
```

This method creates the connection, creating a UDP socket, configuring the client adress and IP

Author

ROC BENAIGES MORAGREGA

3.1.2.3 motorClient()

```
void motorClient (
            void  )
```

This method will be used as a main and will call all the functions and manage the motor of the "client".

Author

ROC BENAIGES MORAGREGA

< ID of the tasks sendData and motorGetPosTask created by taskSpawn()

3.1.2.4 send_data()

```
void send_data ( )
```

This method will be used to send the data (motor position) to the server adress.

Author

ROC BENAIGES MORAGREGA

3.1.3 Variable Documentation

3.1.3.1 buf

```
char buf[32]
```

Buffer of size 32 used to store and then send the data

3.1.3.2 irc_a

```
volatile int irc_a
```

3.1.3.3 irc_b

```
volatile int irc_b
```

3.1.3.4 motorPos

```
int motorPos
```

Position of the motor

3.1.3.5 sockd

```
int sockd
```

Variable for creating a UDP socket

3.1.3.6 srv_addr

```
struct sockaddr_in my_addr srv_addr
```

Variable to configure the client adress

3.2 ircs.c File Reference

This file contain the functions that are common in the client.c and server.c in order to have more organizated all the project. The declaration of the functions is in the "ircs.h" file.

```
#include <taskLib.h>
#include <stdio.h>
#include <kernelLib.h>
#include <semLib.h>
#include <intLib.h>
#include <iv.h>
#include <xlnx_zynq7k.h>
#include <arch/ppc/ppc5200.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <inetLib.h>
#include <sockLib.h>
#include <math.h>
#include "ircs.h"
```

Functions

- void irc_init (void)
- void irc_isr (void)
- void irc_disable (void)

3.2.1 Detailed Description

This file contain the functions that are common in the client.c and server.c in order to have more organizated all the project. The declaration of the functions is in the "ircs.h" file.

Author

ROC BENAIGES MORAGREGA

3.2.2 Function Documentation

3.2.2.1 irc_disable()

```
void irc_disable (
            void  )
```

This method disable the interrupt and registers connected previously.

Author

ROC BENAIGES MORAGREGA

3.2.2.2 irc_init()

```
void irc_init (
            void  )
```

This method inicialize all the registers in order to have it all ready.

Author

ROC BENAIGES MORAGREGA

3.2.2.3 irc_isr()

```
void irc_isr (
            void  )
```

This method connect an interrupt service routine to the hardware IRQ generated by the motor hardware. It uses some definitions from xlnx_zynq7k.h header file, which is a part of the BSP.

Author

ROC BENAIGES MORAGREGA

3.3 ircs.h File Reference

This file contain the declaration of the functions that are common in the client.c and server.c in order to have more organizated all the project.

Functions

- void irc_init (void)
- void irc_isr (void)
- void irc_disable (void)

Variables

- volatile int irc_a
- volatile int irc_b
- volatile int p_irc
- int motorPos
- int recvPos

3.3.1 Detailed Description

This file contain the declaration of the functions that are common in the client.c and server.c in order to have more organizated all the project.

Author

ROC BENAIGES MORAGREGA

3.3.2 Function Documentation

3.3.2.1 irc_disable()

```
void irc_disable (
             void  )
```

This method disable the interrupt and registers connected previously.

Author

ROC BENAIGES MORAGREGA

3.3.2.2 irc_init()

```
void irc_init (
             void  )
```

This method inicialize all the registers in order to have it all ready.

Author

ROC BENAIGES MORAGREGA

3.3.2.3 irc_isr()

```
void irc_isr (
             void  )
```

This method connect an interrupt service routine to the hardware IRQ generated by the motor hardware. It uses some definitions from xlnx_zynq7k.h header file, which is a part of the BSP.

Author

ROC BENAIGES MORAGREGA

3.3.3 Variable Documentation

3.3.3.1 irc_a

```
volatile int irc_a
```

3.3.3.2 irc_b

```
volatile int irc_b
```

Value of the IRC input A and B

3.3.3.3 motorPos

```
int motorPos
```

Value of the position of the motor

3.3.3.4 p_irc

```
volatile int p_irc
```

3.3.3.5 recvPos

```
int recvPos
```

Value of the position received of the motor

3.4 ircs.h

Go to the documentation of this file.

```
1
6 #ifndef _MYLIB_H_
7 #define _MYLIB_H_
8
9 volatile int irc_a, irc_b;
10 volatile int p_irc;
11 int motorPos;
12 int recvPos;
14 extern void irc_init(void);
15 extern void irc_isr(void);
16 extern void irc_disable(void);
17
18 #endif
```

3.5 README.md File Reference

3.6 server.c File Reference

This file represents the server, i.e. the motor that must receive the position of the motor of the client and move to the position received.

```
#include <taskLib.h>
#include <stdio.h>
#include <kernelLib.h>
#include <semLib.h>
#include <intLib.h>
#include <iv.h>
#include <xlnx_zynq7k.h>
#include <lite5200b.h>
#include <arch/ppc/ppc5200.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <inetLib.h>
#include <sockLib.h>
#include <math.h>
#include "ircs.h"
```

Macros

- #define SERVER_PORT 80 /∗ Port 80 is reserved for HTTP protocol ∗/
- #define SERVER_MAX_CONNECTIONS 20
- #define K 25/∗48∗/
- #define MAX_SPEED 0b00000000000000000000001100000000
- #define CW 0b01000000000000000000000000000000
- #define ACW 0b10000000000000000000000000000000
- #define SPIN ∗(volatile uint32_t ∗) (0x43c20000 + 0x000C)

Functions

- void www (void)
- void init_connectionServer ()
- void recv_position ()
- void set_position ()
- void buffer (void)
- void motor ()

Variables

- SEM_ID move_sem
- volatile int irc_a
- volatile int irc_b
- int sockd
- struct sockaddr_in my_addr srv_addr
- struct sockaddr_in my_name cli_name
- char buf [32]
- int addrlen
- int motorPos
- int recvPos
- short dir = -1
- int motorPosBuf [50]
- int recvPosBuf [50]
- int pointerBuf

3.6.1 Detailed Description

This file represents the server, i.e. the motor that must receive the position of the motor of the client and move to the position received.

Author

ROC BENAIGES MORAGREGA

3.6.2 Macro Definition Documentation

3.6.2.1 ACW

```
#define ACW 0b10000000000000000000000000000000
```

3.6.2.2 CW

```
#define CW 0b01000000000000000000000000000000
```

3.6.2.3 K

```
#define K 25/*48*/
```

3.6.2.4 MAX_SPEED

```
#define MAX_SPEED 0b00000000000000000000001100000000
```

3.6.2.5 SERVER_MAX_CONNECTIONS

```
#define SERVER_MAX_CONNECTIONS 20
```

3.6.2.6 SERVER_PORT

```
#define SERVER_PORT 80 /* Port 80 is reserved for HTTP protocol */
```

3.6.2.7 SPIN

```
#define SPIN *(volatile uint32_t *) (0x43c20000 + 0x000C)
```

3.6.3 Function Documentation

3.6.3.1 buffer()

```
void buffer (
            void  )
```

This method stores in a buffer all the data of position of the motor and the received position in order to use them then in the web server.

Author

ROC BENAIGES MORAGREGA

3.6.3.2 init_connectionServer()

```
void init_connectionServer ( )
```

This method creates the connection, creating a UDP socket, configuring the adress

Author

ROC BENAIGES MORAGREGA

3.6.3.3 motor()

```
void motor ( )
```

This method will be used as a main and will create the tasks and call all the functions and it will manage the motor of the "server" and then delate all the tasks.

Author

ROC BENAIGES MORAGREGA

< ID of the task wwwTask created by taskSpawn()

< ID of the task recvData created by taskSpawn()

< ID of the task moveMotor created by taskSpawn()

< ID of the task bufferTask created by taskSpawn()

3.6.3.4 recv_position()

```
void recv_position ( )
```

This method receives the position of the engine from the client constantly.

Author

ROC BENAIGES MORAGREGA

3.6.3.5 set_position()

```
void set_position ( )
```

This method moves the motor in order to the position of the motor and the received position of the other engine are the same. We have the K constant which is the responsible to the quality of regulation (no oscillations, fast response, minimal steady state error, ...)

Author

ROC BENAIGES MORAGREGA

3.6.3.6 www()

```
void www (
             void  )
```

This method creates the simple web server and generates the graphs

Author

ROC BENAIGES MORAGREGA

3.6.4 Variable Documentation

3.6.4.1 addrlen

```
int addrlen
```

3.6.4.2 buf

```
char buf[32]
```

3.6.4.3 cli_name

```
struct sockaddr_in my_name cli_name
```

3.6.4.4 dir

```
short dir = -1
```

3.6.4.5 irc_a

```
volatile int irc_a
```

3.6.4.6 irc_b

```
volatile int irc_b
```

3.6.4.7 motorPos

```
int motorPos
```

3.6.4.8 motorPosBuf

```
int motorPosBuf[50]
```

3.6.4.9 move_sem

```
SEM_ID move_sem
```

3.6.4.10 pointerBuf

```
int pointerBuf
```

3.6.4.11 recvPos

```
int recvPos
```

3.6.4.12 recvPosBuf

```
int recvPosBuf[50]
```

3.6.4.13 sockd

```
int sockd
```

3.6.4.14 srv_addr

```
struct sockaddr_in my_addr srv_addr
```