universidade de aveiro

Institute of Electronics and Informatics Engineering of Aveiro

IRIS Lab

Karma Wheel Chair

Communication Protocol

Synthesis

This document was made to describe the communication protocol between the Joystick control and the motor controller of the Karma wheel chair

Luís Almeida
luisfilipealmeida@ua.pt
Bruno Teixeira
brunodteixeira@ua.pt

Index

Index of Figures

Index of Tables

1 Overview

Our global objective for this initial phase of this project was to try and understand the communication protocol between the joystick controller and the motor controller designed by “PG DRIVES TECHNOLOGY”.

At first we tried to look as much information as possible about all the relevant parts of the wheel chair and found some interesting forums were people explained how they solved the communication between the wheel chair and motor, but none of them tried to decipher the message code, they all “hacked in” the joystick.

The joystick works by having a voltage set point of 2,5 *V*, and has a 9 pin connector that connects to the microcontroller board and converts the signals into one and sends it to the motor controller using one communication bus that has the ability to communicate both ways.

When it is on its stable position the joystick outputs 2,5 *V*, but when you move it that voltage changes to a maximum of 4 *V* or a minimum of 1 *V*, in any direction. This way, the microcontroller present in the joystick board, makes the conversion of those voltages into a binary signal which is then sent to the motor controller.

On the joystick command we also have a few buttons, power on, increase wheel max speed, decrease wheel max speed and a horn. These buttons also send out a signal to the motor controller.

We started by analyzing the signal that was sent in the communication bus and easily deciphered the control message sent from the joystick, which was composed of a 72 bit message. That message was composed of 12 bit frames (6 in total) with one start bit at digital value 0, 8 data bits, 1 parity bit and 2 stop bits.

But we also noticed that after that message there was another, which had a strange shape and was difficult to stabilize. After some tests we found out it was the response from the motor controller to the joystick which made sense, because it had at least to send the battery level for the led display on the joystick command. To properly analyze this message, we had to make some changes to the way we were analyzing the first part, because the response from the motor controller was very weak and the simple fact of using a test probe of the oscilloscope to obverse the signal causes the change in its value.

The response from the motor controller had 77 bits in total. It was composed of 11 bit frames (7 in total) with one start bit at digital value 0, 8 data bits, 1 parity bit and 1 stop bit.

Both messages had digital value 0 at 0 *V* and digital value 1 at 5 *V*. Their descriptions are detailed in the next chapters, as well as the first approach to the communication functions created in order to be able to send a command to the motor controller, without using the joystick command, and to read its response.

2 Description of Protocol

As said earlier we could detect two distinct messages, the Control Message, sent from the joystick command to the Motor Controller, and the Response Message from the motor controller.

The Control Message has 72 bit in total. It is composed of 12 bit frames (6 in total) with one start bit at digital value 0, 8 data bits, 1 parity bit (even parity) and 2 stop bits. Each frame has an approximate period of 310 *µs* which gives an average bit rate of 25.83 *µs* (38,7 *KHz*).

The Response Message has 77 bits in total. It is composed of 11 bit frames (7 in total) with one start bit at digital value 0, 8 data bits, 1 parity bit (even parity) and 1 stop bit. The bit rate is the same as in the control message giving an approximate period for each frame of (284,2 *µs*).

2.1 Control Message

As mentioned before the control message is composed of one start bit, 8 data bits, 1 parity bit and 2 stop bits, so the following characters are the ones we deciphered through our tests corresponding to combination of data bits shown when we pressed a button or moved the joystick.

2.1.1 First Byte

The first byte corresponds to the ID of the Control Message signaling the beginning of a new control message. This byte is always the same:

Table 1 - Control Message - First byts definition

Value	***Function***
0x4A	Control Message ID

2.1.2 Second Byte

The second byte corresponds to the buttons that are pressed in the joystick command.

Table 2 - Control Message - Second byte definition

Value	***Function***
0x00	No button pressed
0x02	Decrease wheels top speed
0x04	Increase wheels top speed
0x20	Buzzer
0xC0	Initiate power off sequence (Press power button)
0x80	Power off sequence (After release of power button)

Each of these values is sent while we are pressing the button in the joystick command. When we release the button it returns to the starting position and the value ***0x00*** is sent to the motor controller.

The **"*Power Off*"** sequence is initiated with ***0xC0*** when we press the Power button on the command joystick and when we release it it continuously sends ***0x80*** until the command joystick powers off.

2.1.3 Third Byte

The third byte has no utilization for the joystick command we have. It could be used in other wheel chair models.

Table 3 - Control Message – Third byte definition

Value	*Function*
00	Standard output

2.1.4 Fourth Byte

The fourth byte corresponds to the movement of the joystick forward and backwards, thus resulting in moving the wheel chair forward and backwards. The position of the joystick is converted to a binary counter (that indicates at zero that the joystick is in the stable position): if we move it forward the counter counts positively, if we move it backwards the counter counts negatively.

When the joystick is full pressed forward the binary counter would represent 100 (***0x64***) and when it is full pressed back it would represent -100 (***0x9C***).

Table 4 - Control Message - Fourth byte definition

Value	*Function*
0x00	No forward or backward movement
0x01 … 0x63 **0xFF … 0x9D**	Binary Counter that represents the speed we want up to the max forward and backwards speed
0x64	Maximum forward speed
0x9C	Maximum backwards speed

2.1.5 Fifth Byte

The fifth byte corresponds to the movement of the joystick right and left, thus resulting in moving of the wheel chair right and left. It works like the fourth byte: if we move the joystick right the counter counts positively, if we move it left the counter counts negatively.

When the joystick is full pressed right the binary counter would represent 100 (***0x64***) and when it is full pressed left it would represent -100 (***0x9C***).

Table 5 - Control Message - Fifth byte definition

Value	Function
0x00	No right or left movement
0x01 ... 0x63 **0xFF ... 0x9D**	Binary Counter that represents the speed we want up to the max right and left speed.
0x64	Maximum right speed
0x9C	Maximum left speed

2.1.6 Sixth Byte

The sixth byte represents the frame checksum. It is calculated as the one's complement of the truncated sum of the previous 5 bytes (i.e, the sum of all 6 bytes is ***0xFF***).

2.2 Response Message

In this section we will present the response message returned by the motor controller. In this message we can determine the occurrence of errors (not yet studied), battery level and current wheel speed velocity level.

2.2.1 First Byte

The first byte corresponds to the first part of the ID of the Response Message signaling the beginning of a new response message. This character is always the same:

Table 6 - First Part of ID Response Message

Value	Function
0xFE	First part of the ID Response Message

2.2.2 Second Byte

The second byte corresponds to the second and final part of the ID of the Response Message. This byte is always the same:

Table 7 - Second Part of ID Response Message

Value	Function
54	Second part of the ID Response Message

2.2.3 Third Byte

On the third byte the 4 most significant bits represent the connection status. If those 4 bits are all zero, the connection is enabled, otherwise it is inactive.

Table 8 – Connection status

Value	Function
0x1X	Connection Down
0x0X	Connection Up

2.2.4 Fourth Data Character

No known uses discovered until the date of the creation of this document.

2.2.5 Fifth Byte

On the fifth byte the 4 most significant bits represent the current battery charge level. It ranges from ***0x00*** corresponding to no battery, to ***0xA0*** corresponding to a fully charged battery.

Table 9 – Battery charge level

Value	Function
0x00	No battery
0x10, 0x20, …, 0x90	Mid charge battery levels. Display has 10 available levels
0xA0	Fully charged battery

2.2.6 Sixth Byte

On the sixth byte the 4 most significant bits represent the Max Wheel Velocity Level. It ranges from ***0x30***, corresponding to the lowest level (Level 1), to ***0xB0*** corresponding to the highest level (Level 5).

Table 10 – Maximum wheel velocity level

Value	Function
0x30	Level 1
0x50	Level 2
0x70	Level 3
0x90	Level 4
0xB0	Level 5

2.2.7 Seventh byte

The seventh byte represents the frame checksum. It is calculated as the one's complement of the truncated sum of the previous 5 bytes (i.e, the sum of all 7 bytes is ***0xFF***).

3 Test Assembly

This project was executed using a PIC32 microcontroller. By disassembling the original controller of the wheelchair we were able to figure out the exact wires that transmitted the information from the analog controller to the motor controller and vice versa. So in order to discover the message patterns and to afterwards test our own program we made a simple circuit using a white board and a few wires and the PIC32.

Next we will show the used setup.

Figure 1 - Original Controller in Off Mode

Figure 2 - Original Controller in On Mode

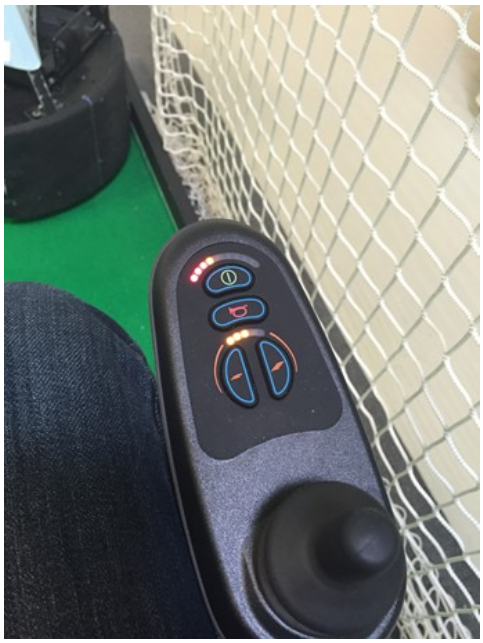

Figure 3 - Original controller - Change speed level

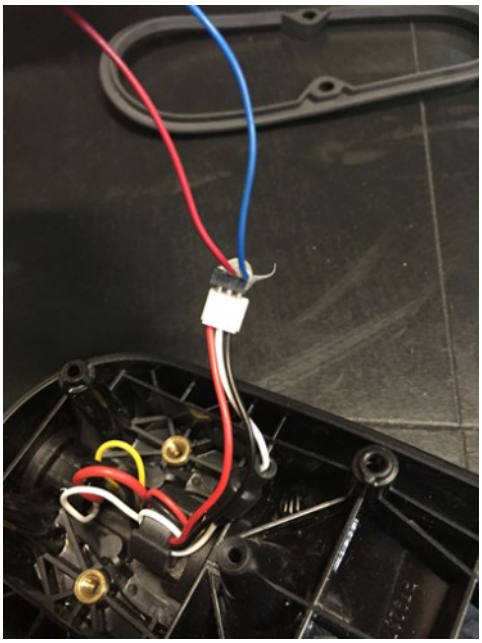

Figure 4 - Opened joystick box

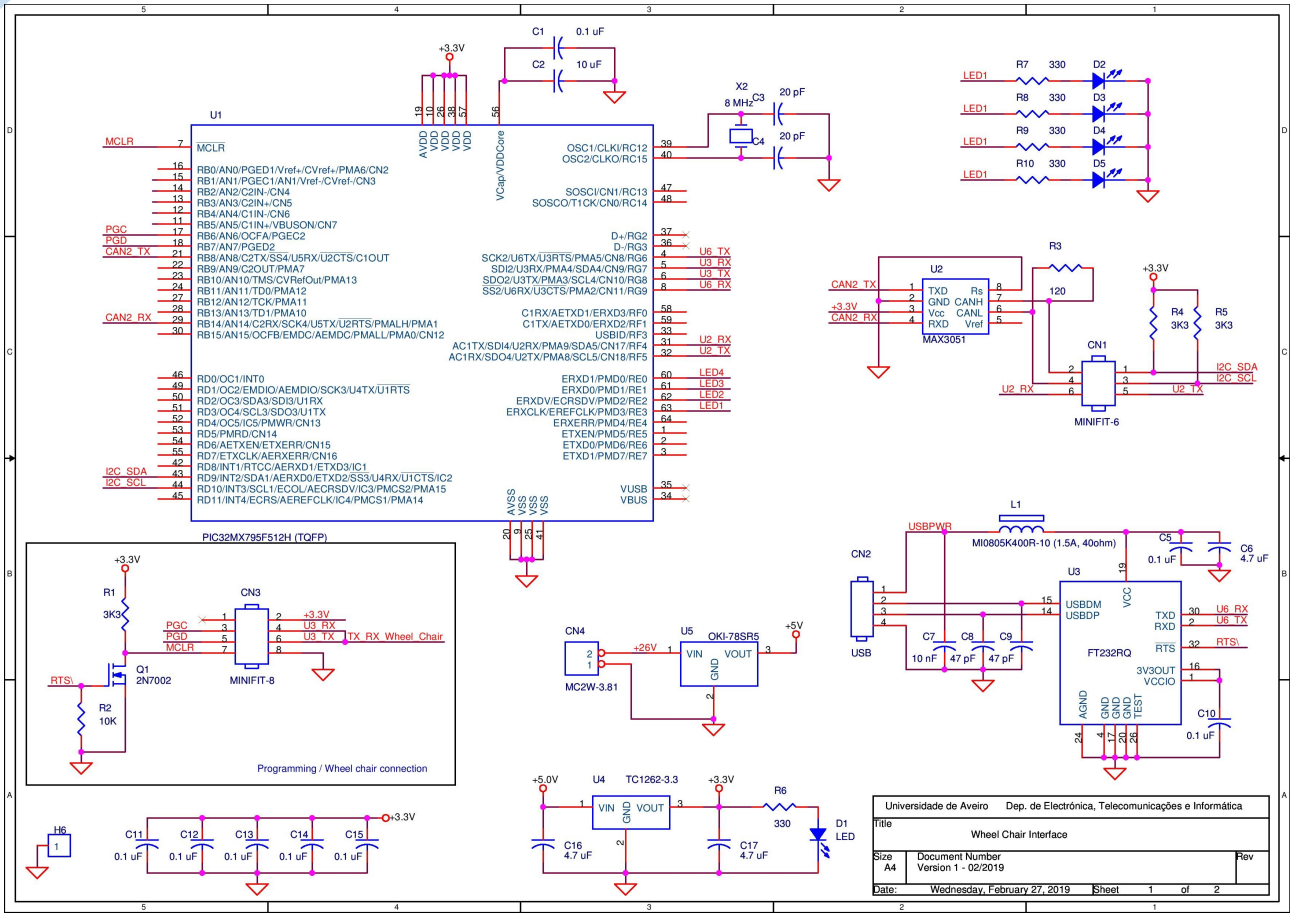

Figure 5 - PIC32 wheel chair interface board

Luís Almeida, Aveiro University
Bruno, Aveiro University