LX-16A Serial Communication

Half-Duplex Serial Communication

To enable the serial half duplex communication with the servo motors, two 74hc126 integrated circuits (ICs) were used to control the RX communication with 3.3V to the esp32 and the TX communication with 5V to the servo motors. The ICs enable the RX and TX line in a sequence to prevent the collision of packets, Figure 1 - Logic gates configuration.Figure 1 show the logic gates configuration for the servo motor serial communication proposed by the manufacture.

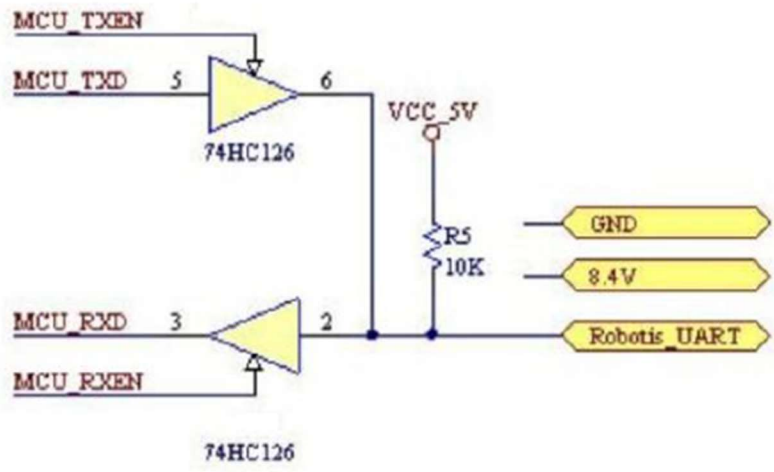

Figure 1 - Logic gates configuration.

According to the proposed circuit to communicate with the servo motors and the logic voltage levels in both ends, esp32 at 3.3V for serial communication and 5V servo motor, an implementation was developed. The circuit follows the same approach but using two 74HC126 with different logic level, 3.3V for RX and 5V for the TX.

Figure 2, shows the implemented approach.

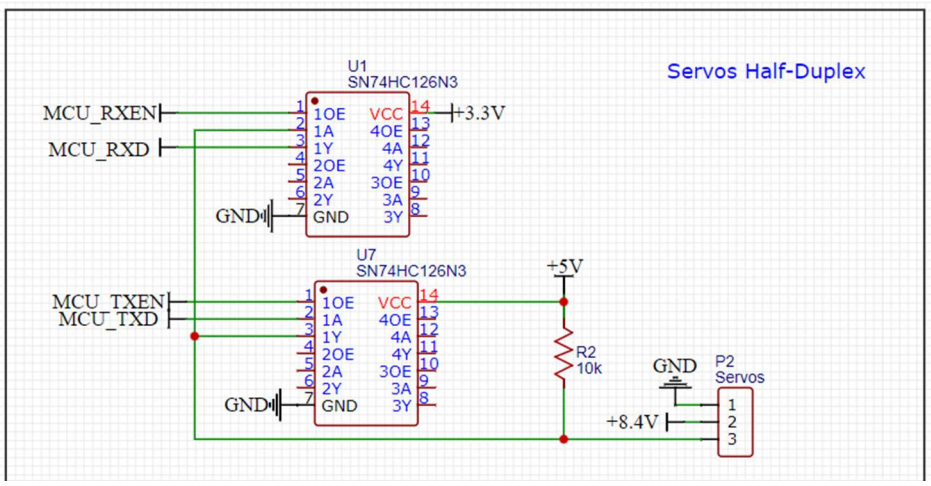

Figure 2 - Implemented circuit for serial communication between esp32 and the servo motor.

Half-Duplex Protocol

The protocol to use when communicating with the implemented circuit is different from the one presented by the manufacture, since the one provided is to communicate through the bus linker v2.2 [2].

First the serial communication follows a 1ms max delay after the Tx transmission for the motor to reply and receive the bytes on the RX line of the esp32 (or another microcontroller).

The communication protocol is an 8bits serial protocol as follows:

MotortimeWrite (0x01) – Send the servo motor to a desire “Pos” in a defined “Time”

- Header – indicate a known message for the motor.
- Servo_id – Id of the motor.
- Msg_len(Servo_id + Msg_len + Param1 …ParamN)) – length of the message except the header.
- Cmd_type – id of the command to perform.
- Pos_LB – Lowbyte of the desired pose.
- Pos_HB - Highbyte of the desired pose.
- Time_LB – Lowbyte of the desired time.
- Time_HB - Highbyte of the desired time.
- CheckSum – Sum of all bytes.

Header		Servo_id	Msg_len	Cmd_type	Pos_LB	Pos_LB	Time_LB	Time_LB	ChekSum
0x55	0x55	0x01	0x07	0x01	Param1	Param2	Param3	Param4	CK

AnglePositionAsk (0x28) – Asks for the servo current position.

- Header – indicate a known message for the motor.
- Servo_id – Id of the motor.
- Msg_len – length of the message except the header.
- Cmd_type – id of the command to perform.
- CheckSum – Sum of all bytes.

Header		Servo_id	Msg_len	Cmd_type	CheckSum
0x55	0x55	0x01	0x03	0x28	CK

AngleOffsetAsk (0x19) – Asks for the servo offset position.

- Header – indicate a known message for the motor.
- Servo_id – Id of the motor.
- Msg_len – length of the message except the header.
- Cmd_type – id of the command to perform.
- CheckSum – Sum of all bytes.

Header		Servo_id	Msg_len	Cmd_type	CheckSum
0x55	0x55	0x01	0x03	0x19	CK

AnglePositionRead (0x28) – Asks for the servo current position.

- Header – indicate a known message for the motor.
- Servo_id – Id of the motor.
- Msg_len – length of the message except the header.
- Cmd_type – id of the command to perform.
- Data – Reply values from servo motors, Value1…ValueN.
- CheckSum – Sum of all bytes.

Header		Servo_id	Msg_len	Cmd_type	Data	Data	CheckSum
0x55	0x55	0x01	0x03	0x28	Value1	ValueN	CK

AngleOffsetRead (0x19) – Asks for the servo current position.

- Header – indicate a known message for the motor.
- Servo_id – Id of the motor.
- Msg_len – length of the message except the header.
- Cmd_type – id of the command to perform.
- Data – Reply values from servo motors, Value1…ValueN.
- CheckSum – Sum of all bytes.

Header		Servo_id	Msg_len	Cmd_type	Data	Data	CheckSum
0x55	0x55	0x01	0x03	0x19	Value1	ValueN	CK

More examples can be seen here:
https://www.dropbox.com/sh/b3v81sb9nwir16q/AABcM4PRPxCYcZ5AFYBzomx8a/LX-15D%20Bus%20Servo/Bus%20Servo%20Communication%20Routines?dl=0&subfolder_nav_tracking=1