This is a part of RoboButler 3000 but might also be useful to anyone else using this type of mobility controller.

The input to the motor controllers is a 4-pin connector which carries power (direct battery connection) and a modified CAN bus called DXBUS.

The joystick unit which plugs into it has an inductive joystick which is good quality. It produces a variable voltage between 0.7V-4.4V. The white and blue wires both produce the same voltage, for one axis, and the yellow and brown wires both represent the other axis.

This plugs into a 7-pin connector on the control board in the joystick module. The pins, reading away from the joystick, are white, blue, brown, yellow, not connected, black (0V) and red (+5v).

Resources

The most comprehensive resource on the controller is a paper by Pedro Daniel Marques and Figueiredo Antao. Unfortunately for us it's in Portuguese: https://ria.ua.pt/bitstream/10773/5643/1/disserta%C3%A7%C3%A3o.pdf. I'll refer to this as the Portuguese document.

A manual from dynamic controls, includes DXBUS pinout on page 25: http://www.dynamiccontrols.com/en/downloads/dx/overview-of-dx-system/74-dx-system-manual/file. Page 34 also shows how to engage flight mode.

Reverse engineering CAN

Using an oscilloscope and a bit of trial and error Jim determined that the CAN bus operates close to 105263 bits/second. So, in the mbed compiler, `can1.frequency(105263);`.

There are two DXBUS ports on the joystick unit so we can tap into the second one to monitor communications while the joystick unit is controlling the power unit. Note that you cannot observe the communication with the motors disconnected

The list of CAN node IDs visible in this conversation are: 8, 64, 170, 700, 701, 668 and 669. It's possible though that 701 and 669 are single-bit errors caused by getting the clock rate wrong.

8 appears to be the power unit and everything else seems to come from the joystick unit.

Node 64 (Joystick unit)

64 changes from a len 3 to a len 7 when the joystick is off centre. When centred it generally transmits [B0][01][20].

When moving forwards it shows:

[B0][01][25][03][FC][02][19]

The fifth and seventh (last) numbers are signed integers which represent the position of the joystick

axis. The fifth integer is right (positive) and left (negative). The seventh is forward (positive) and backward (negative). By looking at the joystick position and data it appears that the acceleration limits are done by the joystick unit. The data on the CAN bus is already damped, so we may be able to bypass the acceleration limits by writing data to the CAN bus.

Other message types from 64: This is a dump of node 64 messages during startup (when the power button is pressed on the joystick unit).

```
Message received: id 64 len 8 data [20][00][00][04][00][80][00][00]
Message received: id 64 len 8 data [20][00][00][04][00][80][00][00]
Message received: id 64 len 8 data [23][00][00][00][00][C0][00][00] # tends
to be C0 if started up in Locked mode, 80 otherwise.
Message received: id 64 len 8 data [23][00][00][00][00][C0][00][00]
Message received: id 64 len 8 data [24][00][00][00][00][54][00][00]
Message received: id 64 len 8 data [24][00][00][00][00][54][00][00]
Message received: id 64 len 8 data [27][00][00][00][00][54][00][00]
Message received: id 64 len 3 data [B0][01][0C]
Message received: id 64 len 8 data [13][01][00][01][00][FF][00][00]
Message received: id 64 len 3 data [B0][01][0C]
Message received: id 64 len 3 data [B0][01][10]
Message received: id 64 len 3 data [B0][01][10]
Message received: id 64 len 8 data [90][01][00][00][06][29][00][00]
Message received: id 64 len 8 data [90][01][00][01][8B][EA][00][00]
Message received: id 64 len 3 data [B0][01][20]
Message received: id 64 len 3 data [B0][01][20]
Message received: id 64 len 3 data [B0][01][20]
Message received: id 64 len 4 data [30][01][00][0E]
Message received: id 64 len 4 data [30][01][00][07]
```

“Locked mode” means the joystick controller is showing “L” on the 7-segment display.

Other nodes

701 transmits [AA][04][20]. the last number rises to 25 when the motors are in motion, whichever direction they're going in.

Simulating the Joystick unit

We can't currently simulate the startup sequence necessary to completely replace the joystick unit. However, we can replace the joystick unit after startup, by quickly switching the can bus from the joystick to the MBED. We can't transmit from the MBED at the same time as the joystick unit as the joystick unit will still be transmitting different values for the joystick.

Using a DPDT switch on the DXBUS and getting the MBED to transmit joystick position (the aforementioned [B0][01][25][03][FC][02][19] message) 20 times a second, the MBED can take over control of the motors. The switch could be replaced by a relay if we can't fully simulate the startup

sequence.

This also only works if the joystick is off-centre to start with - it won't work if the joystick is centred before the switch. There's probably some other initialisation sequence required.

Non-CAN startup sequence

On page 43 of the Portuguese document there's a diagram which shows something required for power-on which is not CAN standard. It appears to show a voltage of 7.5V and 12.4V being applied in a sequence that lasts about 50ms. It's not clear which CAN line this should be applied to or what it's relative to.

Joystick unit

thumb

This is the flexible circuit connector from the Joystick's keypad. From left to right:

- 1 and 7 are internally connected and represent the power button
- 3 is the horn
- 4 is ground (connected to 0V out)

Pressing the power button connects 1&7 to ground (4).

Pressing the horn connects 3 to ground(4).

The horn button connects to pin 23 on the 28-pin PDIP PIC on the side of the 7-segment display.

The two power buttons connect to J2 and another pin next to it.

Category:Projects