

# CAN-PPM Gateway Project Requirements

Goal is to have a very small (shooting for board area of ~4cm x 2cm plus connectors) board which listens to CAN messages and outputs a PPM signal for controlling servos.

Requirements:

- Minimising board size is highly valued; 4cm x 2cm plus whatever footprint is necessary for connectors feels like a good target
- INPUTS:
  - CAN twisted pair
  - 5V power supply\*
  - External power source for servos
- OUTPUTS:
  - At least 2 PPM signals, routed to standard servo connectors
- Servo connectors must be powered directly from an external power rail and capable of supplying 5 A total, but preferably 2-3 A per servo
- This external servo power rail must be capable of being coupled to the 5V input with a jumper or similar mechanism
- At least 2 bits of the CAN ID must be configurable (non-permanently) using jumpers or similar mechanism
- Include a 120-ohm termination resistor which can be enabled using jumpers or similar mechanism
- Must be reasonable to solder by hand (i.e. don't make us solder on a 28-pin surface mount package)
- In addition to whatever main connectors are used, must include some kind of alternative connector which would allow the board to slot into a backplane (e.g. headers, right angle JST, Berg)
- Probably don't need to say this, but please mark sure inputs/outputs are labelled well on the silk screen

Implementation Suggestions:

- I have not been able to find a chip which does this; as far as I can tell, we need to use a microcontroller
- My suggestion (well, ChatGPT's suggestion after some interrogation) is a PIC18F25K83 which is available in a 28-pin DIP package
- Don't forget you'll need a CAN transceiver (and possibly controller depending on your microcontroller, but that PIC18 has one built in); MCP2562 is a good choice, I have a few MCP2542FD samples ordered but might take a long time to arrive
- Open to discussion on connectors, but suggestion would be XT30 for power, screw terminals for CAN
  - Would not be hard to convince me to use a proper locking connector like GX12.
- You can expect to respond to CAN messages in this format for now:

# CAN-PPM Gateway Project Requirements

CAN ID: (29 bits)

Priority [3]	J1939 Parameter Group Number (PGN) [18]														Source Address [8]							
	R	DP	PF			PS (Destination Address) [8]																
X X X	0 0	1 1 1 0 1 1 1 1 0 1 1 0 1 0 A B X X X X X X X X																				

The PF is chosen because that is the only allowed value for proprietary messages (0xEF)

I chose the destination address randomly to reduce the risk of collision  
A and B represent the address bits chosen with the jumpers

Payload: (8 bytes)

Byte Index	Signal Label	Scaling	Limit	Offset	Transfer	J1939 SLOT ID
0	Servo Index	1 count per bit	0 to 250	0	Numeric	129
1	Servo Percent	0.0015625 % per bit	0 to 100.3984375 %	0 %	Numeric	345
2						
3	Don't Care					
4						
5						
6	Message Counter	1 count per bit	0 to 250	0	Numeric	129
7	CRC	N/A	0 to 255	0	Statevalue	93

Signal Descriptions:

Servo Index	<ul style="list-style-type: none"> <li>- Differentiates which servo we want to control</li> <li>- We don't want to set all servos in one message, because then someone wanting to send a new message to adjust a servo needs to know what the correct position for every servo is</li> <li>- For J1939, 0xFF indicates no data available and 0xFE indicates error, so we don't allow either of those</li> <li>- We don't actually need 8 bits for this, but having fields stick to nice byte boundaries makes hand-decoding for verification easier, and we've got space in the packet</li> </ul>
Servo Percent	<ul style="list-style-type: none"> <li>- Pretty straightforward, describes the percentage of the servo range it should be commanded to</li> <li>- At a hardware level, will correspond to the percentage of time in the PPM window before a pulse is generated</li> </ul>
Don't Care	<ul style="list-style-type: none"> <li>- Things are simpler if we always send 8 byte messages</li> <li>- Just here to fill out the packet</li> <li>- Gives room for expansion later without breaking the interface</li> </ul>
Message Counter	<ul style="list-style-type: none"> <li>- Allows us to detect if messages are being lost (for functional safety)</li> </ul>
CRC	<ul style="list-style-type: none"> <li>- CRC8 J1850 checksum of the message, less this field</li> </ul>

# CAN-PPM Gateway Project Requirements

If you have an argument to doing the payload format differently, let's discuss. I would agree that 16 bit precision for the servo percentage is excessive, but maybe 8 bit is too restrictive and we've got space in the packet.

- Bits labeled "X" are allowed to be anything
- You should simply drop any messages which don't have IDs matching this
- We should discuss fault handling:
  - Incorrect CRC [something wrong with stack]
  - Jump of more than 2 in message counter (other than rollover from 0xFA to 0x00) [partial bus/computer failure]
  - No message received for a long time [total bus/computer failure]
  - In a perfect world, we would announce DM1s for these error cases, but perhaps that is excessive

\* We should discuss whether it is worth also supporting 24V input. Background is that arm harnesses have high-current-capable 5V available, and the original purpose was to interface the arm's gripper servo. However, this may now be used on the rover proper for camera control, which does not necessarily have a conveniently located high-current-capacity 5V supply. So, perhaps we should add a 24V input to make things easier.