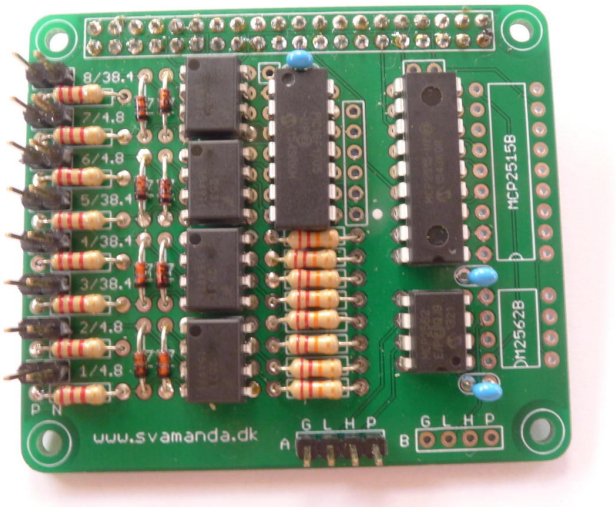
NMEA 0183 and 2000 multiplexer for Raspberry PI

Version 0.1, Draft

by Bjarne Knudsen (bk02@svamanda.dk)



- Four 4,800 baud NMEA 0183 channels
- Four 38,400 baud NMEA 0183 channels
- The NMEA 0183 inputs are optically isolated
- Several NMEA 0183 input options
- One or two NMEA 2000 channels
- Attaches directly to a Raspberry PI

Introduction

The hardware and code in this project allows you to get multiplexed NMEA 0183 input on the serial port of a Raspberry PI. It also allows you to get one or two NMEA 2000 networks connected via MCP2515 chips to the SPI ports of the Raspberry PI. Connecting these data sources to the rest of your network and software is not the purpose of this project. So please don't send me requests about how to do this.

That being said, there are a couple of small programs included that you may be able to use. The code is included which will give you a starting point for your own programs.

This document has four main sections:

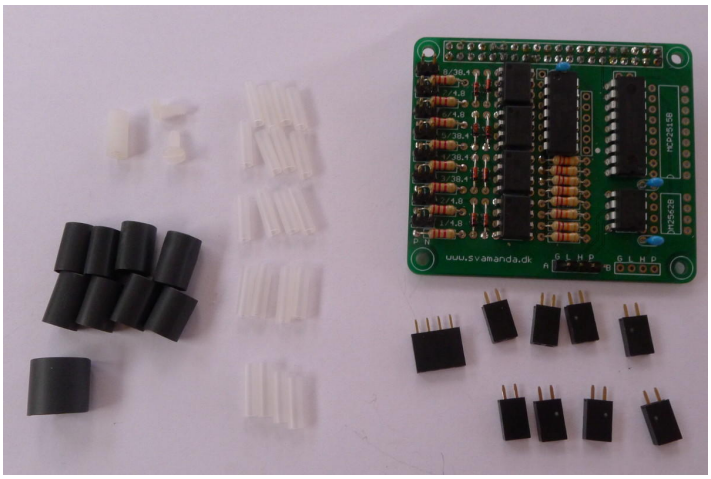
1: The short version: How to set things up if someone provided the board for you

2: The Raspberry PI setup: How to set things up on the Raspberry PI

3: The long version: More details including how to change various settings

4: Building your own: How to build you own multiplexer

If someone was nice enough to build you one of these, your kit should look something like this:



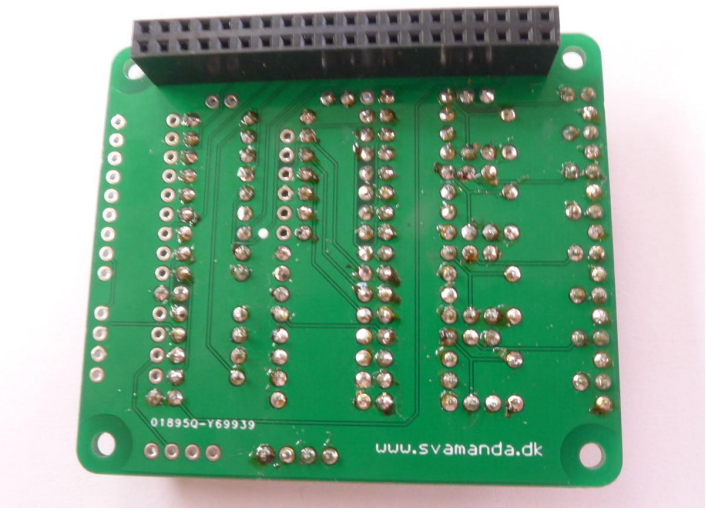
1 The short version

1.1 License

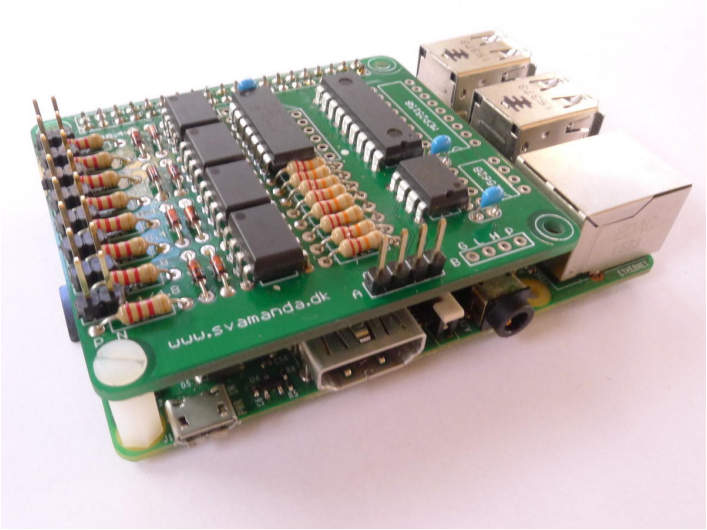
You are free to do whatever you want with everything in this project, but I am not responsible for anything you do with it.

1.2 Attaching inputs

The board is attached to a Raspberry PI using the 40 pin connector:



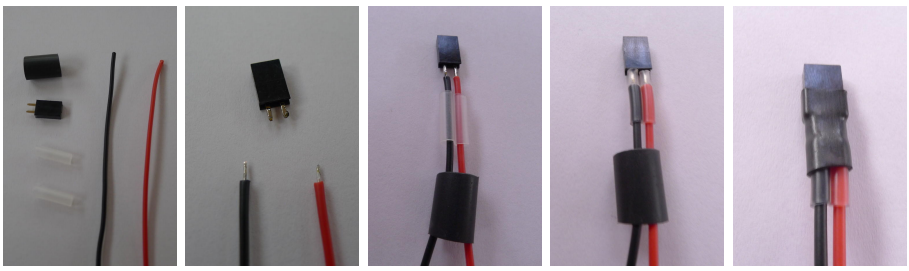
At least one spacer should be attached to keep the board from touching the Raspberry PI:



You can see a spacer on the left in the foreground.

The are eight NMEA 0183 input channels. The baud rate and channel number is marked on the board next to the connector for the channel (see image on front page). Four are 4,800 baud and four are 38,400 baud. The positive lead is the one nearest the edge of the board ("P" and "N" marks the positive and negative at the end of the row of connectors). No damage is done if the inputs are reversed, but no signal will be received. Input voltages from 5 to 15 volts are accepted. The inputs are optically isolated according to the NMEA standard.

The inputs are connected using female pin headers. Here is a way to solder leads to the female headers using the heat shrink:



Presolder the header and the leads, put the leads through the heat shrink, then solder. After this, shrink the clear heat shrink first and then the black. This approach is used for both the NMEA 0183 and NMEA 2000 inputs.

The NMEA 2000 inputs are marked "G", "L", "H", and "P" which is ground, low data, high data, and 12 V, respectively. The multiplexer is

powered from the Raspberry PI, so the ground level of the Raspberry PI and the NMEA 2000 network must be the same (this should be the case if you power the Raspberry PI with a 12 V to 5 V converter from the ship's batteries. Since the multiplexer is not powered from the NMEA 2000 network, ground and 12 V from the NMEA 2000 network are not connected. This means that reversing the connector should not cause any harm. I still attach the ground and +12 to the four pin header to make the connection more mechanically robust.

1.3 NMEA 0183 multiplexer

The multiplexed output is received on the built in serial port on the Raspberry PI. The speed on this connection is 115,200 baud. This means that if all inputs on the board are receiving data at full speed, some sentences will be lost. This will be quite unusual in real life. Even if sentences are received on all inputs at the same time, the data are buffered, so if there is a lull in the input data after this, the buffered sentences will be sent.

Channels 6–8 are special in that input on channel 8 suppresses input on channel 7 which in turn suppresses input on channel 6. This is useful when you have several GPS sources and you only want one of them to be present on the output channel. One possible setup is:

Channel 8: NMEA input from an AIS transponder which includes GPS position.

Channel 7: NMEA input from secondary GPS instrument.

Channel 6: NMEA input from tertiary GPS instrument.

When the AIS transponder is on, that data is output but none of the other GPS data. If the AIS transponder is off, the secondary GPS instrument is used. When both of those are off, the GPS position from the tertiary GPS instrument is used. The suppression is constantly being evaluated, so if the AIS transponder is turned off, the secondary GPS data will start being output after 2.5 seconds.

If you are interested in more options, read the long version.

1.4 NMEA 2000 input

Typical boats only have one NMEA 2000 network, so the normal board has just one NMEA 2000 channel. If needed, there is room for a second NMEA 2000 channel on the board. This could be for a second NMEA 2000 network

or another CAN bus network such as some networks of tank level sensors and/or battery monitors.

2 The Raspberry PI setup

2.1 Setting up the serial port

The serial port on the Raspberry PI is used as a login terminal by default. To make it a normal serial connection, type:

```
sudo raspi-config
```

Choose "Boot Options", then "Advanced Options", then "Serial" and finally "No" to the question of whether the login shell should be accessible over serial. Exit after this, but do not reboot yet.

Open the file `/boot/config.txt` and change this line:

```
enable_uart=0
```

To this:

```
enable_uart=1
```

Then reboot the Raspberry PI. The serial port should now be ready. To set up the port for communication with the multiplexer, enter:

```
stty -F /dev/ttyAMA0 115200 -echo
```

You can now receive data with:

```
cat /dev/ttyAMA0
```

2.2 The Raspberry PI software

More details will follow.

- There will be software for sending NMEA 0183 data to other computers using UDP and TCP.
- There will be a program to read NMEA 2000 data with basic functionality for translating some sentences to NMEA 0183.

3 The long version

3.1 NMEA 0183 multiplexer

3.1.1 More on the data flow

An input sentence is a series of up to 80 ASCII characters with values between 32 and 255 (both included) followed by any number of ASCII characters with values of 31 or less. Typically, the ending characters will be a return and a newline or just a return. But a zero character or a tab will for example also end a sentence.

Any sentences with over 80 characters are discarded in their entirety. Note that the ending characters does not count towards these 80. NMEA 0183 sentences will have at most 80 characters according to the standard, so they should never be too long. If a sentence takes too long to receive, it will also be discarded (i.e. if around 13 to 20 seconds passes during its reception).

No matter how a sentence is ended on the input, its output will end with a return and a newline (or optionally just a newline, see below). The multiplexer forwards all ASCII sentences, not just NMEA 0183 sentences. This means that the multiplexer can also be used for other input types such as data from a Navtex instrument.

The output order is the same as the input order measured at the time the first ending character is received. Inputs are always buffered before being forwarded to the output. This means that there will always be a short delay between the time when a sentence is fully received on an input to the time it is fully transmitted on the output. If there is no other sentences to wait for, this will be about 7 μs for a 80 character sentence at an output speed of 115,200 baud.

There are 11 banks in the PIC for storing input sentences. This means that if multiple sentences are received at overlapping times, they will typically not be lost, just delayed a little extra. If sentences are received at the same time on all four 38,400 baud inputs, five such sets of input sentences in a row without any delays are required before a sentence is lost.

If all four 4,800 baud inputs are receiving at the same time as the 38,400 baud inputs, a second set of simultaneous sentences on the 38,400 baud inputs will result in a lost sentence. So sentences can be lost, but it will typically be a rare occurrence.

3.1.2 Setting options

If data is sent from the Raspberry PI to the board over the serial connection, the NMEA 0183 multiplexer goes into interactive mode where settings can be changed. This is indicated by "Interactive" being output from the board on the serial connection and forwarding of inputs being stopped. In the interactive mode, commands are sent to the multiplexer using the serial connection.

Note that there must be a period of about two seconds without input in the serial connection before new input makes the multiplexer go to interactive mode. This helps ensure that the interactive mode is not entered by accident.

Open two terminals on the Raspberry PI and type this in one:

```
cat > /dev/ttyAMA0
```

And this in the other:

```
cat /dev/ttyAMA0
```

You can now enter commands in the first terminal and see the output in the second. Just press enter in the first terminal to go to the interactive mode.

Each command is one line of input. Here is an overview of the commands:

Description	Format	Parameters
Channel numbers	C(b)	b = 0: no, 1: yes
Fast channels	F(h)	h = fast channels as hex bits
Return newline	N(b)	b = 0: newline, 1: return and newline
Invert inputs	I(hh)	hh = inverted channels as hex bits
Suppression	U(c)(hh)	c = channel, hh = other channels as hex bits
Discard	D(c)(hh)	c = channel, hh = hex ASCII char: 0 or $\geq$ 32
Output baud	B(s)	s = 0: 4,800, 1: 38,4000, 2: 115,200
Input level	H(hh)	hh = Schmitt channels as hex bits, others TTL
Print settings	P	
Save settings	S	
Load settings	L	
Restore factory	R	
Debug info	G	
Exit	X	

The "X" command ends the interactive mode and "Done" is output from the multiplexer before it starts forwarding the input again. The parentheses are not included in the input, so "D821" means that sentences starting with the

character "!" (ASCII 33 = 0x21) are discarded on channel 8 (see below for more information). All letters must be uppercase, also in the hexadecimal numbers.

If a wrong command is given, the multiplexer outputs "Error".

3.1.3 Channel numbers

Using the "C1" option, each output sentence will start with a single digit indicating which channel it was received on. This is useful for de-multiplexing the input on the Raspberry PI. You can for example attach a Navtex instrument as an input and store that data in a separate place while not sending it to programs expecting NMEA 0183 data. This option is turned off by entering "C0".

3.1.4 Fast channels

All the 38,400 baud input channels can be configured as 4,800 baud instead. This is done using the "F" option. It is followed by a single hex digit which is a bit mask for which of the four fast channels should in fact be 38,400. The bits refer to input channels 3, 4, 5 and 8 with the least significant bit referring to channel 3. So "F0" makes all channels 4,800 baud, while "FF" makes all fast channels 38,400 baud.

3.1.5 Return newline

By default, the output sentences are always ended by the return character followed by the newline character. This can be changed to just newline by entering "N0". The default behavior is "N1".

3.1.6 Invert inputs

If one of the input channels receives input that is inverted, use the "I" option. It is followed by a two digit hex number that is a bit mask for which channels should be inverted. Inverted input could for example come from an USB to serial adapter connected to a computer.

3.1.7 Suppression

This command is used to suppress the input on one channel depending on whether there is input on other channels. The channels to look for input on

are given as bits in a two digit hexadecimal number. So "U5C0" means that the input on channel 5 is suppressed if there is input on channel 7 or 8.

If a channel is suppressed due to other input, this channel is considered to have input when deciding whether other channels should be suppressed. This is used in the default setting which is "U780" and "U640" meaning that channel 7 is suppressed if there is input on channel 8 and channel 6 is suppressed if there is input on channel 7. In this case input on channel 8 will suppress channel 7 but also channel 6 whether or not there is input on channel 7.

3.1.8 Discarding sentences

For each channel, it is possible to discard sentences based on their first character. This will typically be used to discard AIS data from a particular channel by discarding sentences starting with "!". For discarding AIS data on channel 8, use option "D821". This means that on channel eight, all sentences starting with ASCII code 0x21 ("!") are dropped. To avoid discards, enter ASCII code zero.

3.1.9 Output speed

The default output speed is 115,200 baud. It can be set to 4,800, 38,400, or 115,200 baud using the "B" command. These speeds are encoded as 0, 1, 2, and 3 respectively. When the output speed is changed it happens when the interactive mode is left. The input speed is also changed to the same value.

3.1.10 Input levels

By default the code on the PIC use Schmitt triggers for input voltages levels from the optocouplers. The "H" option allows TTL voltage levels to be used instead. The channels to use Schmitt triggers are given as a bit mask in a two digit hex number.

With the 3.3 V supply voltage given to the PIC, Schmitt triggers have 0.66 V as low and 2.64 V as high. TTL has a low of 0.80 V and a high of 2.00 V. If you find an instrument where the NMEA input is not working, you can try changing these input levels.

3.1.11 Print settings

The "P" command prints all settings. The output format is the same as the commands to set these settings.

3.1.12 Save settings

The "S" command saves the current settings so when the Raspberry PI is powered off and on again, these settings will be in use.

3.1.13 Load settings

The "L" command loads the last saved settings or the factory settings in case no settings were ever saved.

3.1.14 Restore factory settings

The "R" command restores the factory settings.

3.1.15 Debug output

Using the "G" command, some debug information is output:

```
VE 0.1
RI 2002
FE 00
CG 00
LO 00
SL 00
T1 0D05
T2 0D05
```

The information is:

VE: Code version

RI: Revision ID of the PIC

FE: Number of sentences dropped due to frame errors (i.e. a stop bit was not high as it should be).

CG: Number of sentences dropped due to congestion (i.e. not enough storage space)

LO: Number of sentences dropped due to being too long

SL: Number of sentences dropped due to being too slow

T1: The minimum number of cycles for a pass through the main loop

T2: The maximum number of cycles for a pass through the main loop

All the output is in hexadecimal format. The counters overflow and restart at 256. The main loop is supposed to take 3,333 instruction cycles (0xD05). If the two cycle counters are not both equal to this number, it is due to a programming error.

All counters and timers are reset when exiting the interactive mode.

3.2 NMEA 2000 input

NMEA 2000 channel A is connected to SPI channel 0 on the Raspberry PI. Channel B is connected to SPI channel 1. It is not only possible receive NMEA 2000 data, but also to send it from the Raspberry PI if someone was to write the appropriate software. The interrupt from channel A is connected to GPIO 5 (pin 29) on the Raspberry PI and the interrupt on channel B is connected to GPIO 12 (pin 32).

4 Building your own

4.1 License

I am not responsible for anything you do with this board and accompanying code.

You are free to do whatever you want with everything in this project, including the Raspberry PI code, the micro controller code, and the hardware. But if you want to make me extra happy consider doing one or more of these things:

- Let me know if you find a clear bug that you can tell me how to reproduce.
- Let me know if you have made improvements to the code.
- When you are building one of these for yourself, build two and give one away (maybe in return for dinner and/or beer).
- Start a business building these and sell them at a reasonable price.

4.2 Parts

Note that it is important to use the MCP2562 for the physical NMEA 2000 interface. The reason is that the logic is 3.3 V instead of 5 V. This capability is not supported by MCP2561 or MCP2551.

Both the capacitors on the NMEA 2000 part of the board should be included even if only one NMEA 2000 channel is included.

Here is a list of all needed parts for a board with one NMEA 2000 channel:

Number	Description	Source	Part number
1	PCB	Seeed	Order using zip file
1	PIC16F1705	Mouser	579-PIC16F1705-I/P
1	MCP2562	Mouser	579-MCP2562-E/P
4	HCPL-2531	Mouser	512-HCPL-2531
1	MCP2515	Futurlec	MCP2515
8	2.2 kΩ resistor	Futurlec	R0022K14W (10)
8	22 kΩ resistor	Futurlec	R022K14W (10)
3	100 nF capacitor	Mouser	594-K104K15X7RF53L2⋆
8	diode	Futurlec	1N4148 (10)
1	2×20 female pin header	Futurlec	FHEADD40
1	4 pin female header	Futurlec	FHEADS4 (10)
8	2 pin female header	Futurlec	FHEADS2 (10)
1	4 pin male header	Futurlec	HEADS4 (10)
8	2 pin male header	Futurlec	use HEADS4 above†
8”	3/16” clear heat shrink	Mouser	5174-13323 (48”)
3.2”	1/4” black heat shrink	Mouser	5174-1141 (48”)
0.4”	3/8” black heat shrink	Mouser	5174-1381 (48”)
1	Nylon spacer	Mouser	761-2056-440-N
2	Nylon screw	Mouser	534-9327

⋆I have not personally used this part, but it should be ok.

†For the two pin male headers, just split the four pin ones.

If two NMEA 2000 channels are needed, add one MCP2562, one MCP2515, one set of four pin headers, and a bit of heat shrink.

All the parts except for the PCB can probably be bought from Mouser if you prefer to get it all from one place.

4.3 Soldering

All the parts are through hole components, so soldering is easy. The holes surrounded by a square box are for pin headers. The large 2×20 header is a female header that is soldered on the back side of the board. The other headers are male headers. The single and dual headers right next to the

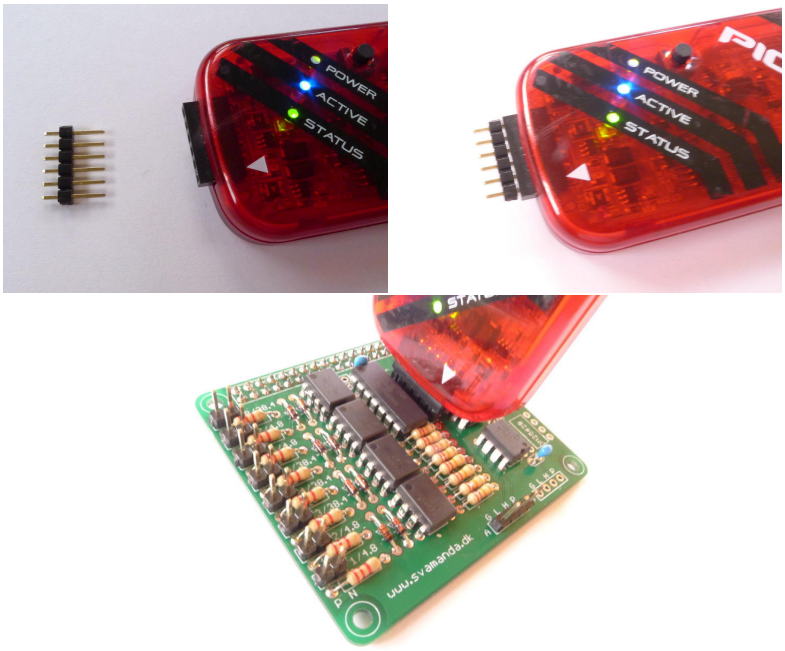
chips are generally not used. The six pole header next to the PIC need not be attached either (see how to program the PIC below).

The three dual holes with a rounded frame are for 100 nFcapacitors. Resistors are shown with their symbols and values. Diodes are shown with their symbols.

If you are worried that some of the chips may have faults or that you overheat them during soldering, you can get sockets from Futurlec, solder those to the board, and attach the chips when you are done soldering.

4.4 Programming the PIC

The PIC is programmed using the PicKIT3 programmer. It can be bought at Mouser (part number 579-PG164130) . The six holes in a row on the board next to the PIC is the connection for the programmer. The white dot indicates pin 1 which is marked with an arrow on the programmer. I generally don't solder a header on here, I just put male header pins in the programmer and push it against the board:



The software for the programmer is Microchips Integrated Programming Environment which is downloadable from their site. You just have to download the programmer part of the software.

The hex file is included in this project and is called `nmea.hex`. If you want to make your own hex file, it is made using `gputils`. The code and Makefile is included.

4.5 Versions

4.5.1 Only NMEA 0183

A board without any MCP2515 and MCP2562 chips may also do without the two 100 nF capacitors next to these. Such a board will work as an NMEA 0183 multiplexer alone.

4.5.2 Only NMEA 2000

Skipping all the components except the MCP2515, MCP2562, and their two capacitors will make a board that is NMEA 2000 only, except for one issue: The MCP2515 chips need a clock input. Normally this comes from the PIC. So one option is to also add the PIC and its capacitor.

Another option is to get the clock from the Raspberry PI. To do this, solder a connection between the two holes at the end of the MCP2515 for channel A. This connects GPIO 6 (pin 31) on the Raspberry PI with the clock inputs on the two MCP2515 chips. Use software to output a suitable clock (8 MHz is what the PIC provides) on this pin on the Raspberry PI. In this case, the PIC and its capacitor is not needed.

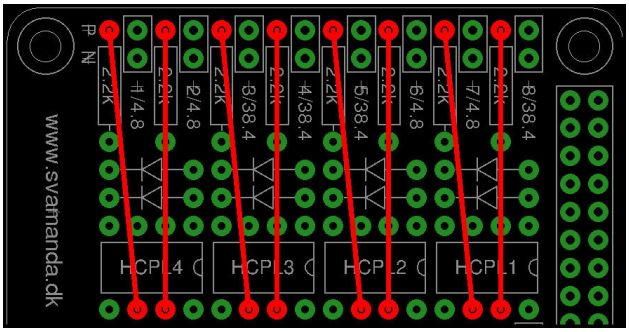
4.5.3 Fewer NMEA 0183 inputs

If you are sure you don't need all the NMEA 0183 inputs, you can skip some of them. Just decide which of the optocouplers you are going to leave out and leave out all the components between them and their corresponding NMEA 0183 inputs. You can also skip their corresponding 22 kΩ resistors which are placed on the board in the same order as the inputs.

Since the 38,400 baud channels can be configured as 4,800 baud, it makes sense to skip channel 1 and 2 which are 4,800 baud only.

4.5.4 For the cheapskates

If you are a real cheapskate, you can drop all optocouplers, the 2.2 kΩ and 22 kΩ resistors, and the diodes. Instead, just connect a 22 kΩ resistor for each channel from the hole at the edge of the board to the corresponding output from the optocoupler (the two middle pins on the far side are the outputs):



There are a few downsides to doing this. One is that there will be no optical isolation. The other is that the ground level must be the same for the inputs as it is for the Raspberry PI. This will usually be the case, so this trick generally works. Needless to say, such a board does not fulfill the NMEA 0183 standard. The inputs are still protected from reverse voltage. In this setup, the negative connector of the NMEA 0183 inputs is unconnected, so no ground loops will arise even without optical isolation.

Another effect of doing this is that the inputs to the PIC will be inverted. This can be fixed using the option for inverting inputs described above.

The advantage is price: the optocouplers are high quality and their cost is more than half of the total price of the board.