

CAN/isoSPI Filtering Deep-Dive 26-27

Goal:

We would like to decrease noise and increase reliability across our CAN bus and isoSPI lines.

Overview:

What circuit is being studied here? What are early suspicions about the issues we've been seeing in this circuit? Are there any concepts/theories that we're missing?

Resources:

- [Controller Area Network Physical Layer Requirements - TI 2008](#)
- <https://www.ti.com/lit/ta/ssztam0/ssztam0.pdf>
- [TCAN1044AEV CAN Transceiver Datasheet - TI](#)
- [PESD1CAN TVS DIODE Datasheet](#)
- [CSS Electronics Can Bus Explained](#)
- [Bulletproof isoSPI](#)

Deep-Dive:

CAN:

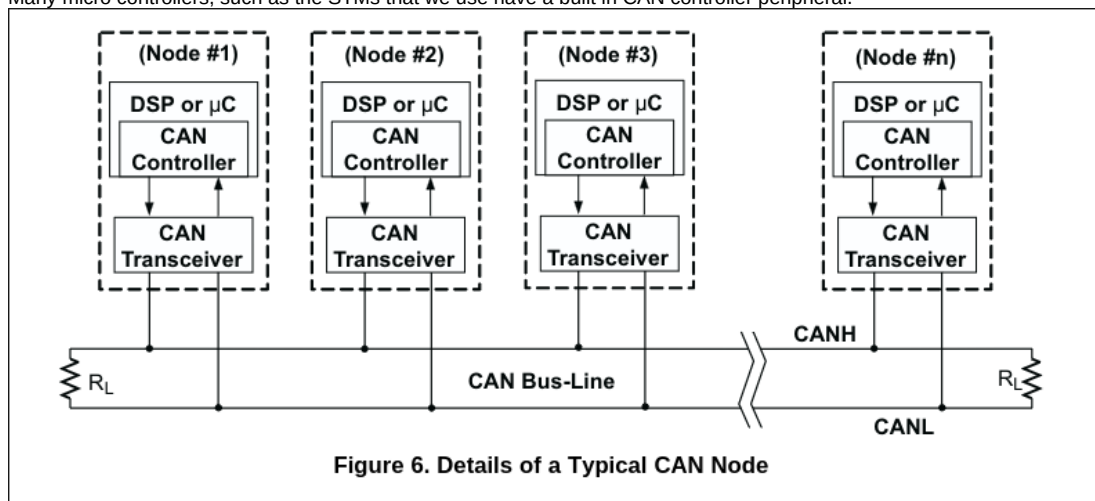
What is a CAN Bus?

Controller Area Network (CAN) is a communication protocol originally developed in the 1980s for automotive application and standardized in 1993 under ISO 11898. The CAN protocol calls for a differential paired bus on which nodes engage in half duplex communication. Notable features include non destructive bit-wise arbitration, error detection, and clock synchronization across devices on the bus.

Physical Details:

A CAN bus consists of devices (nodes) connected by a twisted pair of wires labeled CAN High (white) and CAN Low (white with a blue stripe). In the most common topology, a bus, Nodes are tapped off of the line on stubs, whose lengths must be minimized for signal integrity. At the ends of each bus, the network is terminated with a 120 ohm resistor to absorb reflections. There are other possible topologies such as loop, star, and full networks which all have their own benefits and drawbacks.

Each node on the bus consists of 3 main components: the Microprocessor which processes data and decides what to send, the CAN Controller which handles the datalink layer to convert the data into valid CAN frames, and the CAN transceiver which drives the voltages to send signals down the harness. Many micro controllers, such as the STMs that we use have a built in CAN controller peripheral.



Since CAN is a differential paired signal, signals are not read as a low or high voltage, but rather the difference between the CAN High and CAN Low lines. This rejects EMI as the two lines are coupled and close to each other so any noise will be almost exactly the same on both lines and therefore the difference will be constant. This is further compounded by single terminated shielding on our CAN wires which helps to limit interference from motors and inverters. CAN protocol defines a recessive 1 as both lines being passively driven to their resting voltage (~2.5V on the TCAN1044AEV) and a dominant 0 as CAN High driven high (between 2.75 and 4V) and CAN Low driven low (between 0.5 and 2.25V). This means that if two nodes try to write to the bus at the same time the 0 will win out because a 1 does not source or sink current whereas a 0 does.

Since all nodes need to be able to send and receive messages to and from all other nodes, the timing will not be preserved exactly equally due to propagation delay. Therefore, the CAN Transceivers sample later into each bit (usually 70-90%), so that errors due to early or late rises will be avoided, and information inside of the messages are used to synchronize the clocks.

Datalink Specifications:

First the Start of Frame(SOF) writes a dominant zero to the bus to tell all the controllers that a message is about to be sent.

Following that, an 11 bit identifier is sent along the bus(or 29 if you use the 29 bit identifier). The identifier denotes the importance of the can message, and a lower identifier means the message has more priority. If two messages are sent at the same time, the one with the lower identifier will win due to Physical Non-Destructive Arbitration(PNDA). As each bit of the identifier is sent, the CAN node also reads back each bit. If the bit is a 0 but that node intended to send a 1, it knows that another node must have sent a dominant 0. The CAN Controller will then listen to the rest of the higher priority message and then attempt to send its message again afterwards.

After arbitration has been solved, the RTR bit will denote if the node is sending a message or making a remote request. In the case of a remote request, other controllers will write the proper data to the bus. In the normal case, the CAN node writes its data. The IDE bit denotes if there will be another 18 bits for the extended identifier. r1 and r0 bits are reserved for future protocol expansion

When sending Data, the DLC field denotes how many bytes of data will be sent in the data section. Following the DLC the data is actually sent.

After data is sent and received, the can controller conducts a cyclical redundancy check to ensure that there was no error in the transmission. The transmitter calculates a checksum before sending the message and sends it in the CRC the receiver(s) calculate their own checksum as they receive the message and compare it to the transmitted CRC. If there is any discontinuity, an error is thrown. Following the Checksum, the receiver(s) acknowledge the message with the ACK bit and the the frame ends.



Figure 3. Standard CAN: 11-Bit Identifier



Figure 4. Extended CAN: 29-Bit Identifier

Image of Arbitration in action:

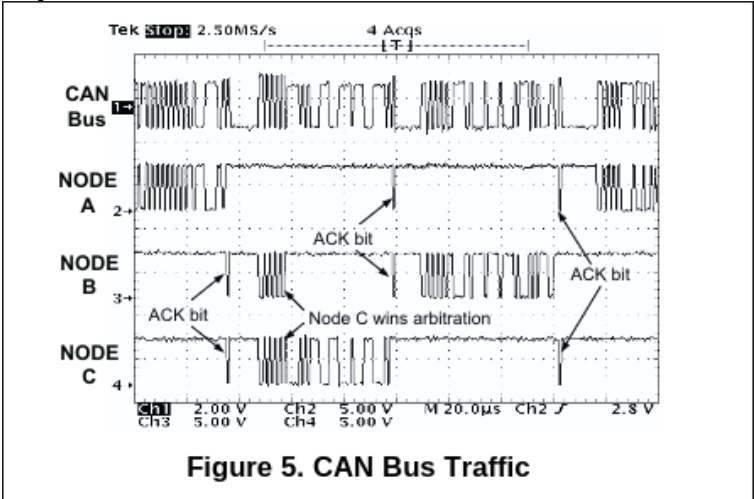


Figure 5. CAN Bus Traffic

Signal Integrity Considerations:

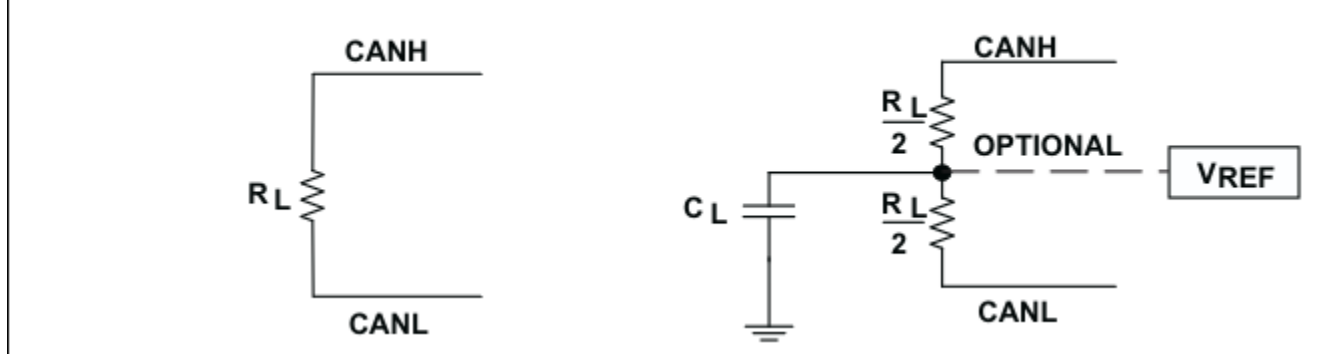
Because the act of signals traveling down a CAN line cause propagation delay, the maximum bus length is determined by the baud rate or vice-versa. The rule of thumb for this is to assume a propagation delay of 5ns/m and keep Signaling Rate (Mbps) × Bus Length (m) 50. Considering our CAN lines are generously 5m long, we could run our bus at 10Mbps which exceeds the typical maximum CANFD Baud of 8Mbps.

Similar to the maximum length of the bus, the length of the stubs must be limited to ensure reflections are attenuated before a bit is sampled. To ensure that the noise is rejected along with a margin of safety, stubs should be limited to 1/3 of the critical length of the bus(propagation delay of the wire*switching time). The TCAN1044AEV chip we use has a differential rise time of 50ns and fall time of 28ns and the propagation delay of our wire is defined by the dielectric coating the conductor(ETFE: 5.4ns/m) so our worst case critical length is 5.19m leading to an absolute maximum stub length <= 1.72 meters. Although that is the Theoretical maximum, standards recommend significantly lower stub lengths ~0.3 meters for performance at Mbps because of inconsistencies in cabling, tolerance stacking with switching and sampling, and stubs acting as transmission lines.

Our transceivers have a 28ns fall time so the knee frequency of the signals is 17.85Mhz(the maximum frequency that carries relevant power from the signal). Since the signals travel through our wires at 185000000m/s(see propagation delay in prior segment) we have a wavelength of 10.36m. Since anything above 1/10 of the wavelength must be treated as a transmission line, 1m of wire would be the maximum lumped circuit and anything above that would start to cause ringing and other issues.

CAN lines must be terminated with 120 ohm resistors to match the bus impedance and absorb reflections. To increase noise rejection it is possible to use a split termination. This still acts as a 120 ohm resistor on the end of the line, but the capacitor acts like a low pass filter for common mode noise. The terminations do not increase bus capacitance because the differential signal has the same difference from resting voltage at both high and low, so the capacitor will always "see" the resting voltage plus common noise.

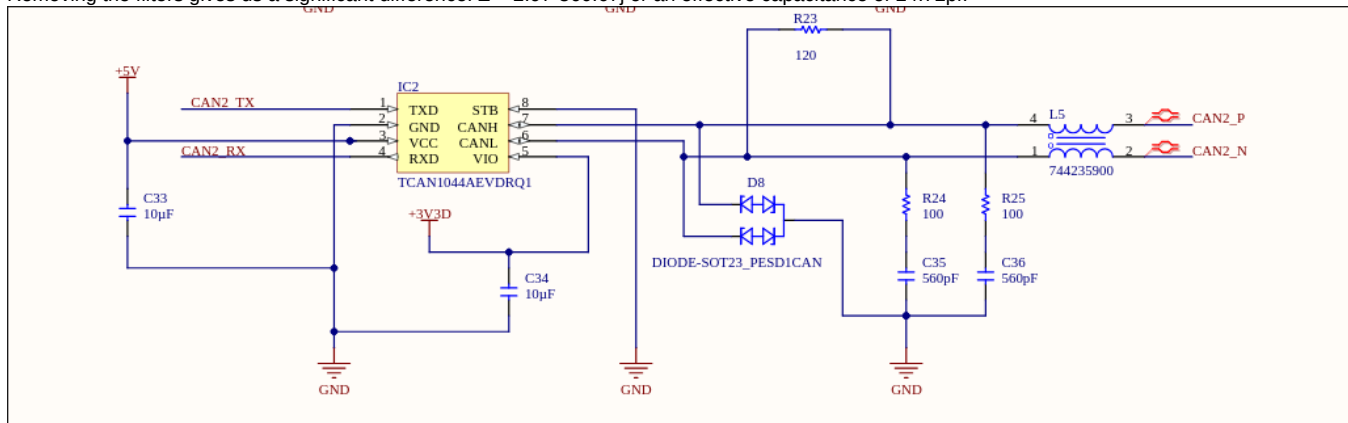
Split Termination



Empty CAN lines can be modeled with constant impedance $Z = \sqrt{L/C}$ L and C being per unit length. As CAN nodes are added, their capacitance lowers the local impedance and can lead to impedance mismatches and reflections across the bus. Assuming that the impedance on any area of the bus with nodes has an impedance of Z' , you can calculate the reflected voltage. In the worst case, during arbitration or acknowledgement, when a can line transitions from dominant to recessive, you have errors when $Z' < 0.71Z$ (See [Controller Area Network Physical Layer Requirements](#) 4.11 for derivation). We can then approximate Z' in terms of the distance between nodes to find the minimum distance between nodes.

The CAN node comprises of a transceiver, common mode choke, tvs diode, low pass filters, pcb traces, connector, and can line stub. The impedance of this at our highest relevant frequency(17.85Mhz) is as follows: ($Z_{\text{filter}} = 200-31.8j$ $Z_{\text{trace(capacitance)}} = -2972j$ $Z_{\text{transceiver}} = 12.2-891.8j$ $Z_{\text{TVS}} = -1049j$ $Z_{\text{Connector}} = -8916j$) all in parallel in series with $Z_{\text{wire(.3m)}} = .016+33.6j$ $Z_{\text{CMC}} = -.05+.05j$ $Z_{\text{trace(resistance+inductance)}} = .5+2.02j$ yields 141.10-59.08j or an effective capacitance of 158.99pf.

Removing the filters gives us a significant difference: $Z = 2.97-360.67j$ or an effective capacitance of 24.72pf.



Our lumped capacitance of a can node can be approximated as the sum of the capacitance of the can transceiver, TVS Diode, CMC, Low pass filters, traces, connector pins, and wire stub. Using PDB-CAN2 as an example, we have transceiver differential capacitance of 10pF, TVS capacitance of 8.5pF, CMC capacitance is not listed, but since we are at such low frequencies relative to its rating it is reasonable to assume negligible capacitance, we have 2 low pass filters each with a 560pF capacitance in series(canhi to ground, ground to canlow) leading to 280pF capacitance, trace lengths to connectors are 105mm and per online calculator that gives a 3pF capacitance, pin capacitance can be assumed to be ~1pF, and a .3m stub yields a capacitance of 15pF. This leaves us with a total differential capacitance of C of 317.5pF.

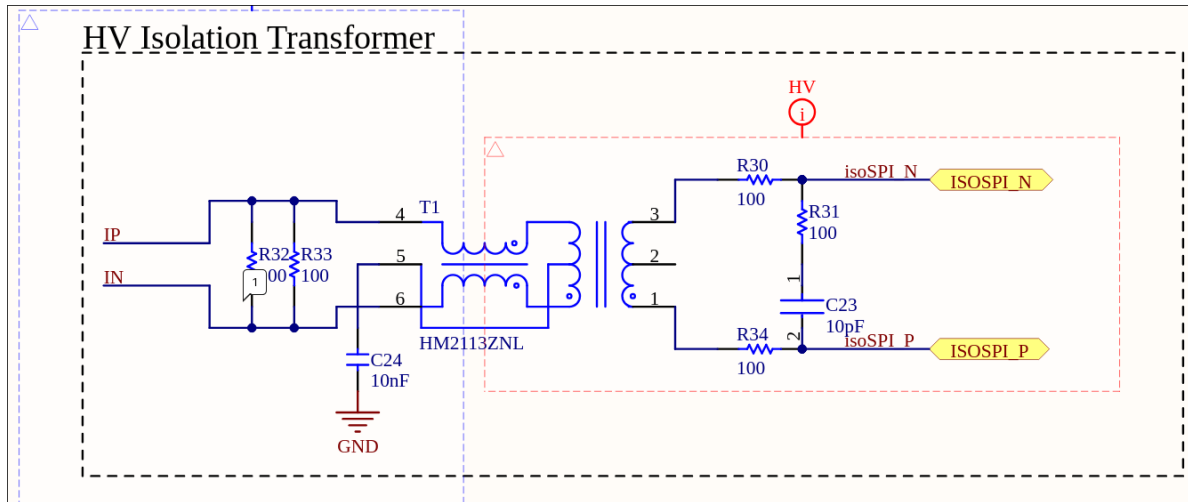
From these calculations we find the minimum distance between nodes to be $317.5 / (.98 * 50) = 6.479$ meters. If we were to drop the low pass filters in favor of split termination, the minimum distance would be .77m and cutting stub length from .3m to .1m would bring minimum node distance to .561m

isoSPI:

What is isoSPI?

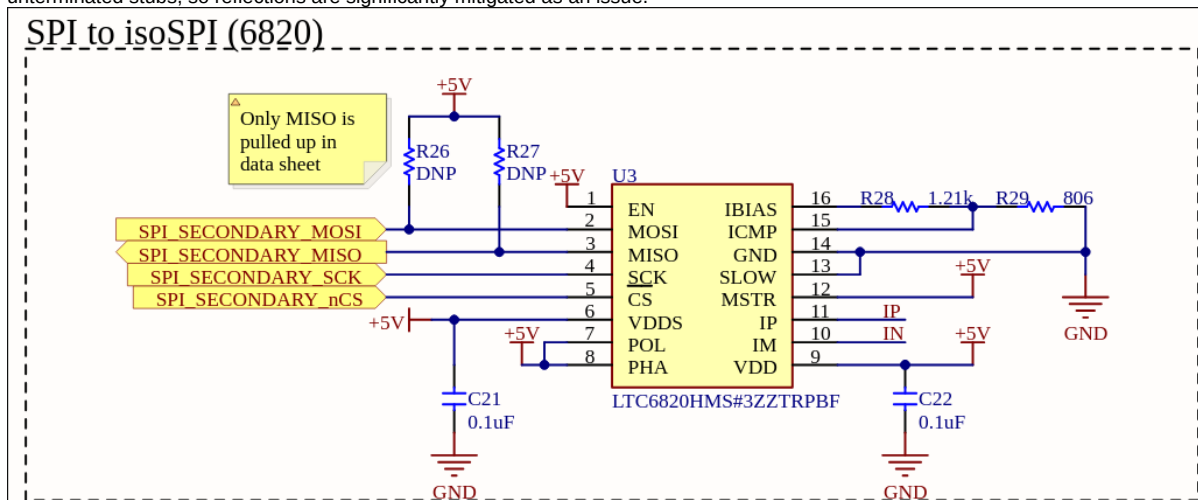
IsoSPI is a proprietary communication protocol developed by Analog Devices. It takes a regular 4 line SPI bus and it serializes it into a 2 line half duplex differential signal which is isolated through a transformer. IsoSPI ICs can be daisy chained, so after the 6820 converts a spi signal into an isoSPI signal, then you can just connect each 6811(or6813) BMS chip in series.

In our implementation of isoSPI we use multiple LT-6811-2s in a multidrop configuration(just like CAN). To improve signal integrity on these lines, we can do the exact same things we want to do on CAN lines: minimize stub lengths, introduce split termination, ensure that impedance is consistent to mitigate reflections. Additionally, you can center tap the pulse transformer to help with noise rejection as seen with C24 on our schematic.



The Problem With our Architecture:

Although we can make these minor changes to our isoSPI topology, we have made an oversight in our implementation of isoSPI. The automotive standard is to use LTC6811-1s in our use case. These versions of the 6811 support daisy-chaining which eliminates many of the signal integrity concerns we see with a multidrop topology. First of all, the impedance is consistent across the line since there are no capacitive nodes to drop the impedance and no unterminated stubs, so reflections are significantly mitigated as an issue.



Daisy chaining does have drawbacks, namely each chip introduces a delay as it relays information down the chain. A singular 6811 has a worst case delay of 300ns, so down and back over 12 chips we would see a 7.2 microsecond delay on the bus. This means we cannot safely run a spi clock rate of above 138khz, but that is safely above the 10khz sample rate that we use on our isoSPI bus. The other issue is that losing one chip makes us lose all chips further down the line. We can solve this solution by adding a second 6820 at the end of the isoSPI chain and run that back to the STM to create a loop, so we are tolerant of any one connection failing.

Other Possible Considerations

Changing the Signal to Noise Ratio by changing the IBIAS and IMCP pins is a way to improve signal integrity. In our case, the 1.21k and 806 ohm resistors lead to a bias current of .99mA which is right under the 1mA maximum so we have already pushed this to the limit of the 6811's specs

I have seen some pages arguing that the common mode choke should be on the wire side of the pulse transformer rather than the transceiver side. The rationale seems to be that excessive common mode noise could saturate the pulse transformer and cause signal losses or other issues, but information is conflicting. All analog devices implementation sheets that I could find show the orientation that we use and many forums and write-ups support this topology, or similar ones that use differing components to act as the isolation. At most, this seems like a minor consideration that could be looked at further if we come across more evidence for switching, or someone with more expertise identifies a failure mode within the pulse transformer, but it does not seem relevant at the moment.

Conclusion:

CAN:

- stub lengths need to be minimized to preferably $<0.3\text{m} / 1\text{f}$ to minimize signal reflections
- We can implement split termination to help attenuate noise without impacting the impedance on the lines
 - The capacitance of the rc filters that we have on each of our nodes may be enough to lower the impedance near the nodes and increases signal rise time, so replacing them with a split termination should increase integrity
- Impedance control on PCB traces is not necessary as they are significantly shorter than wavelengths, but it should be kept in mind
 - ensure continuous ground planes below the traces
 - avoid vias if possible, and if not make sure both high and low have them and keep them close to the transceiver
- While it is more convenient for us to put our termination resistors into the CAN schematic, it may be beneficial to moving to a separate termination on the bus. This means that even if we lose whichever boards were previously terminating our can lines, we still have live CAN lines which would be good for logging
- can traces should cross power traces at a 90 degree angle whenever possible
- If there is still ringing or other noise that drops out near the end of the signal, we can push back when we sample each bit to allow more time for noise to be attenuated. This would be tested by looking at the CAN signals on a scope and viewing where the noise is

IsoSPI:

- If we continue to use a multidrop bus, the same considerations we take for CAN could be taken on the isoSPI lines. Namely, split termination and calculating the proper node spacing for impedance continuity
- The better solution, although it requires more design work, seems to be to move to a daisy chain rather than the topology we use now