Diagnosing Numerical Pathologies using a Modern Digital Oscilloscope

Roland Gamper

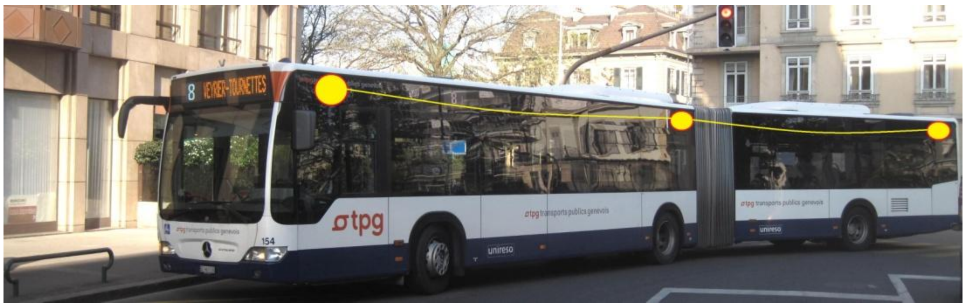
Introduction

This paper explains and documents the usage of ProtoBusMAG, a software option, now available on all LeCroy oscilloscopes. This case study spans three functions: MessageToValue, DeltaMessage and BusLoad. Each one is exemplified on real data on a CAN bus emitted by an inertial sensor.

ProtoBusMAG operates on any decoded stream generated by a LeCroy oscilloscope, such as UART, MIL-STD-1553, SPI, ARINC 429, LIN, FlexRAY, MIPI, etc. and provides unprecedented insights into the quality and structure of the data at both the analog and digital levels.

Context

Geneva's public Transportation Authorities (TPG) has initiated a project aimed at improving passenger's comfort on board their vehicles. To that effect, the Laboratory of Numerical Systems (LSN) of Geneva's University of Applied Sciences has been contracted to develop acquisition modules capable of measuring accelerations in 3 directions, with great accuracy and at the appropriate rate. These small modules will be deployed on board TPG's vehicles, allowing real time measurement of the acceleration values, during travels on Geneva's transportation network. The setup allows a precise monitoring of the shaking, vibrations and centrifugal forces perceived by the passengers. Furthermore, the correlation with the GPS position of the vehicle yields valuable and objective information helping to improve critical spots on the network infrastructure. The same system can also be used to train the drivers, especially on long vehicles, where the trailing end might still be in a curve.



.

Figure 1: Typical TPG bus, with a 3 Node sensor's network distributed along the vehicle.

Implementation Details

The core acquisition module consists of an LIS3LV02DQ inertial sensor (ST Microelectronics), slaved to a LPC2292 microcontroller (NXP), over an I2C bus. Each module is capable of measuring 3D accelerations with a 15 bit resolution, in a range of ±6[g].

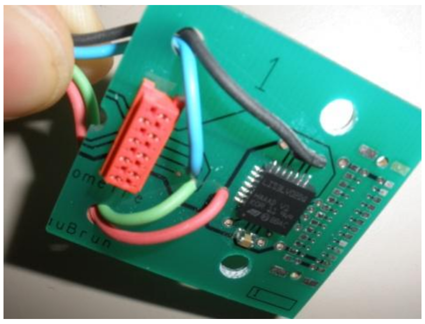


Figure 2: The LIS3LV02DQ inertial sensor mounted on a small PCB is the heart of the acquisition system, and delivers the necessary data to assess passenger's comfort in real time.

The microcontroller polls the inertial sensor at a rate of 15 ms and broadcasts the X, Y and Z accelerations onto a CAN bus operating at 250 kb/s, interconnecting the sensors of the vehicle (as shown in Fig. 1).

A ruggedized PC taps the data from the CAN network and logs it onto a storage device. The system also interfaces to a GPS sensor, gathering the positional information (WGS84/CH1903) necessary to interpret the inertial data at post processing time.

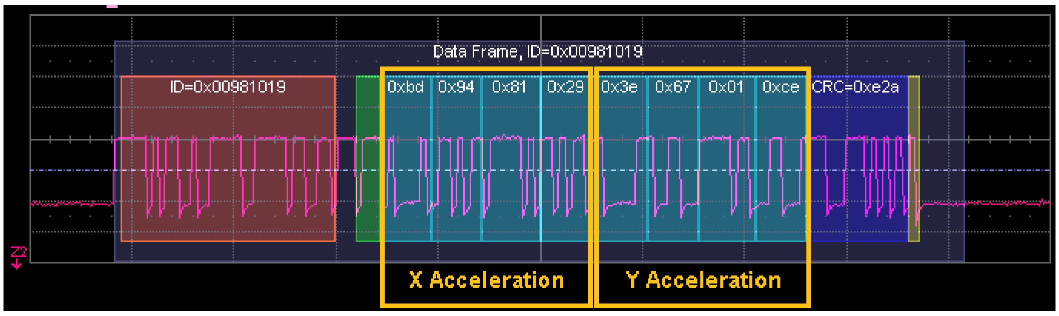


Figure 3: Typical CAN Frame 0x981019, containing X and Y accelerations expressed as IEEE754 floating point numbers. The message length is about 125 bits (with slight variations due to the insertion of the stuff bits), and spans a time of 500 µs since every bit is 4 µs long.

When in operation, the system continuously measures acceleration data in 3 directions every 15 ms as well as positional data every second and broadcasts this data in real time on the same CAN bus, in the floating point 32 bit format shown in Fig. 3.

CAN Traffic Topology

The message structure and distribution outlined above leads to the overall CAN traffic shown below. Each module is unaware of the presence on the bus of the other modules, and emits its data at an even rate, but asynchronously to other modules. A smart time slot allocation rule, based on the CAN prioritization scheme among the sensors yields a balanced distribution of each sensor's data. The validation of this mechanism is described below.

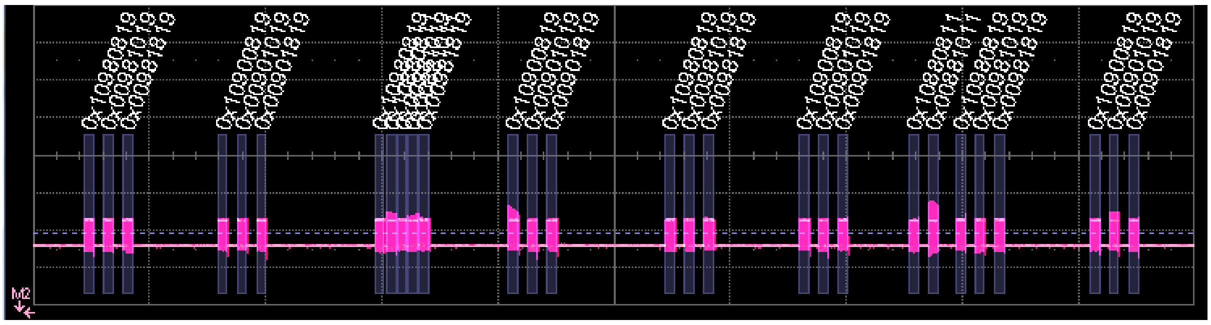
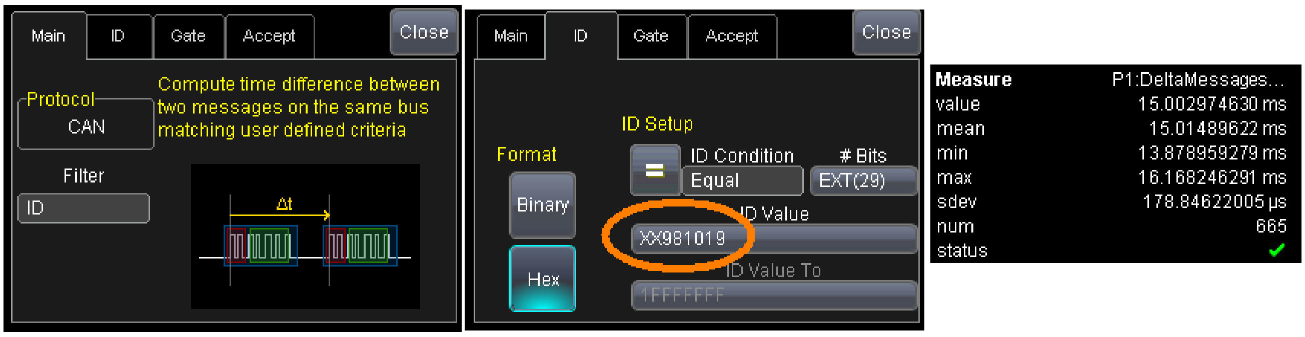


Figure 4: Snapshot of the overall CAN traffic on the CAN bus (250 kb/s) of a moving vehicle.

Validation of the Data distribution on the bus, using DeltaMessage and BusLoad

As a first validation item, LeCroy's ProtobusMAG was used to verify the even distribution of the relevant data onto the bus. This is best done using the Function DeltaMessage, which computes the time elapsed between messages with the same ID on the CAN bus. In this particular case, we focus on message 0x981019 by filtering it out and using a long record of 10 seconds to obtain a statistically meaningful result. As can be seen in the results section of the image below, 665 occurrences of the message are spaced in average by 15.01 ms, with minimum and maximum intervals of 13.9 and 16.2 ms, which is fully acceptable for this type of mission.



Protocol: CAN

Compute time difference between two messages on the same bus matching user defined criteria

Filter: ID

ID Setup

Format: Hex

ID Condition: Equal

Bits: EXT(29)

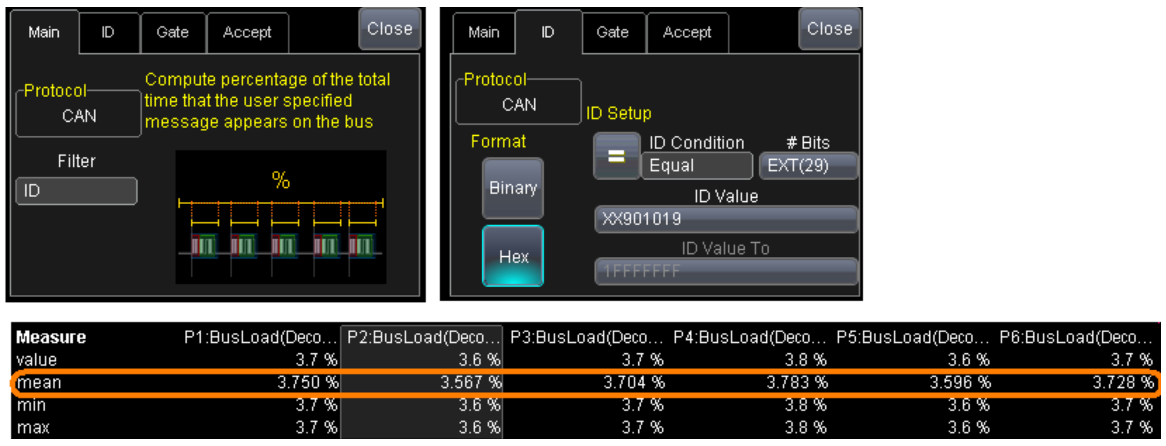
ID Value: XX981019

ID Value To: 1FFFFFFF

```
Measure      P1:DeltaMessages...
value          15.002974630 ms
mean            15.01489622 ms
min            13.878959279 ms
max            16.168246291 ms
sdev            178.84622005 µs
num                        665
status                       ✓
```

Figure 5: The DeltaMessage Parameter settings (left) and results (right) for monitoring CAN ID 0x981019 on the bus, shows a mean distance of 15.01 [ms].

Another critical value for the system design engineer is the balanced broadcast of all 3 sensors on the bus. These numbers are easily extracted from the traffic by use of the BusLoad parameter.


Main | ID | Gate | Accept | Close

Protocol: CAN

Compute percentage of the total time that the user specified message appears on the bus

Filter: ID

Main | ID | Gate | Accept | Close

Protocol: CAN

ID Setup

Format: Binary | Hex

ID Condition: = Equal

Bits: EXT(29)

ID Value: XX901019

ID Value To: 1FFFFFFF

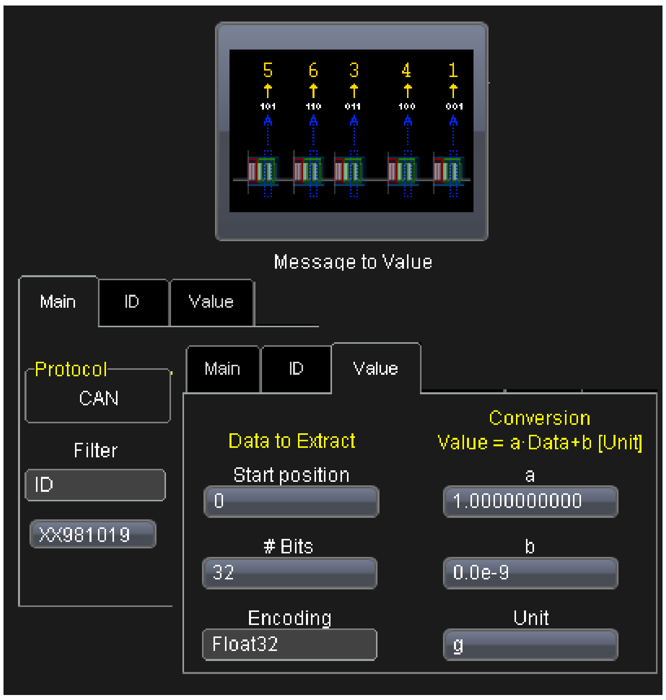
Measure	P1:BusLoad(Deco...	P2:BusLoad(Deco...	P3:BusLoad(Deco...	P4:BusLoad(Deco...	P5:BusLoad(Deco...	P6:BusLoad(Deco...
value	3.7 %	3.6 %	3.7 %	3.8 %	3.6 %	3.7 %
mean	3.750 %	3.567 %	3.704 %	3.783 %	3.596 %	3.728 %
min	3.7 %	3.6 %	3.7 %	3.8 %	3.6 %	3.7 %
max	3.7 %	3.6 %	3.7 %	3.8 %	3.6 %	3.7 %

Figure 6: The BusLoad parameter, with its setting and results, used on 6 of the messages, establishes that each message uses about 3.7 % of the bus bandwidth. This correlates with the fact that a message of 500 [µs] is broadcasted every 15 [ms], therefore a busload of 3.3 %. The extra 0.4% is explained by the fact that a complete message length encompasses the CAN Inter Message Gap after the last bit.

Detecting numerical pathologies in the data

Observation Setup

The observation setup is matched to the example shown in Figure 3, and spans the first 32 bits of CAN message 0x981019 as a floating point number, with units in [g] to reflect the fact that we observe accelerations. Units will propagate throughout the computational chains, to the Parameters, the Track, the Cursors, the Descriptors and all subsequent processing elements. The unit propagation helps in keeping a clearer mental model.



Message to Value

Main | ID | Value

Protocol: CAN

Filter: ID

XX981019

Main | ID | Value

Data to Extract

Start position: 0

Bits: 32

Encoding: Float32

Conversion
Value = a·Data+b [Unit]

a: 1.0000000000

b: 0.0e-9

Unit: g

Figure 7: Setup of the MessageToValue parameter for the X acceleration, a 32 bit subfield of message 0x981019, located at bit 0. (See also Fig. 3)

Healthy case

Figure 8 shows the normal graph of the 3 D accelerations expected on the CAN bus using the MessageToValue parameter. This is the reference, when looking at pathological cases later.

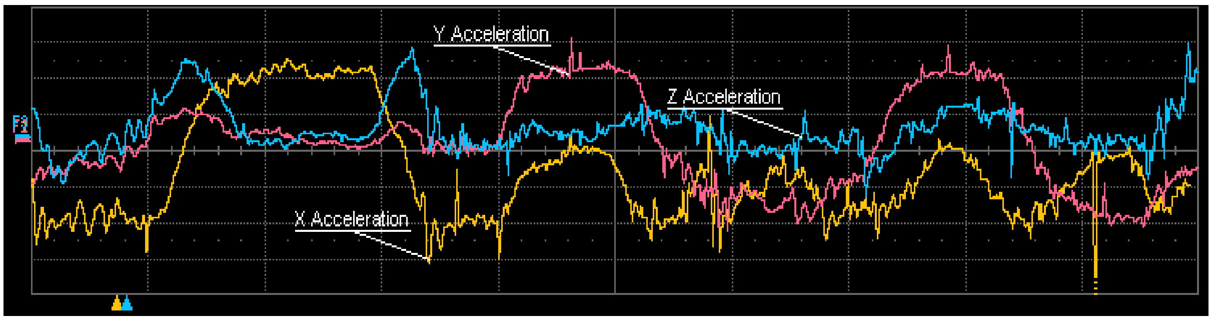
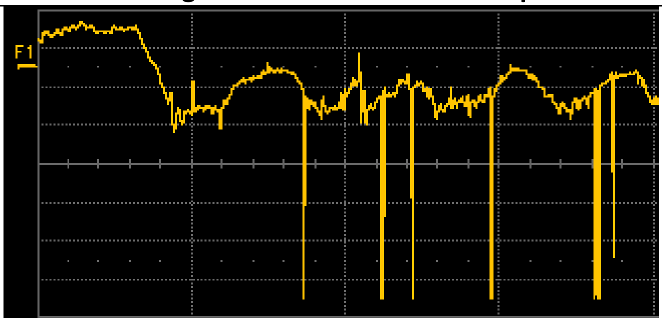
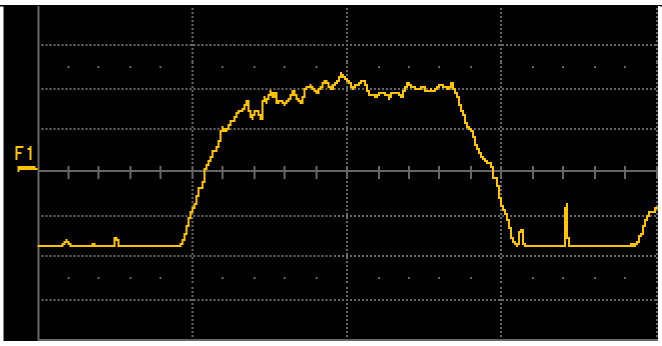
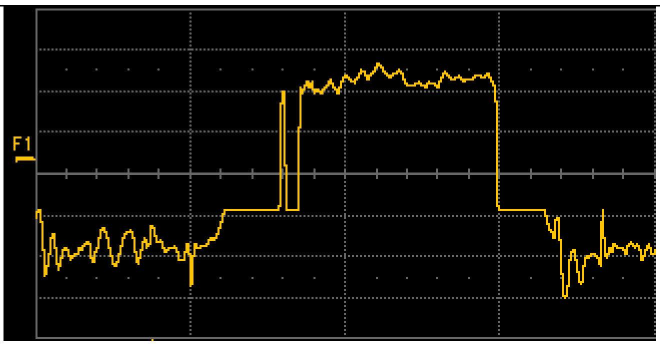
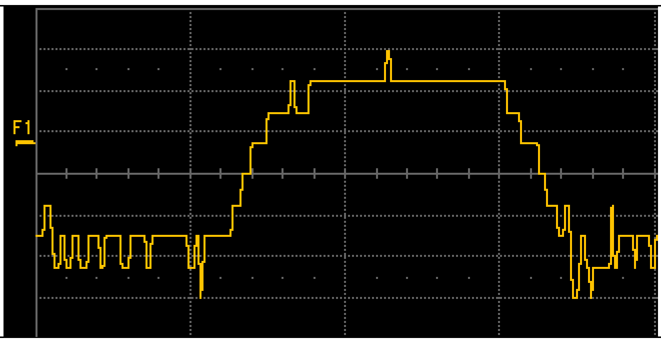


Figure 8: This is the normal 3D output of the LIS3Lv02DQ inertial sensor when manipulated softly. Accelerations remain between ±1 [g] in all directions.

Pathological cases

The following 4 examples make use of the MessageToValue parameter on the acceleration data broadcasted on the CAN bus. The test procedure is rather simple and based on manual, smooth rotating and tilting of the sensor shown in Fig. 2. The normal response of the sensor is shown on Fig. 8, with the typical X-Y-Z acceleration components shifting between themselves. The following table shows what appears on the oscilloscope screen, in the Track signal, when various pathologies affect the numerical processing chain.

Track of MessageToValue on the oscilloscope	Observation	Possible Cause
	Glitches or discontinuities on signal, beyond the possible accelerations expected in a smooth manual test.	**Hardware** errors in the sensor. **Software** conversion errors. **Transmission** errors on either digital bus.
	Railing effect on the low side of the signal.	**Hardware** faults in the ADC or the inertial sensor. **Software** errors such as an incorrect cast in C, incorrect byte swap, incorrect mask, partial transfer on the I2C bus.
	Banding effect, a value range of the output is skipped.	**Hardware** errors in the sensor. **Software** error in the micro controller code. **Transmission** errors on digital bus.

	Stair casing effect	Loss of least significant bits, either in hardware or in software.

Other possible observations

It would also be possible to conduct the same observations on the I2C bus between the µC and the inertial sensor. The MessageToValue function is also used, albeit with different setup, since the values are 16 bit signed instead of floating points.

The DeltaMessage function could be used to measure the conversion delay between the messages on the I2C bus and the corresponding message on the CAN bus. This measurement reflects the fact that the microcontroller acts as a gateway between both digital buses.

If available, the analog output of the sensor could be observed. It would yield a curve similar to those of Figure 8, with a slight forward lag in time, since the analog data precedes its apparition on the digital buses. Here the AnalogToMessage function could be used.

It would also be possible to sum all 3 acceleration vector's components using the Function set offered by the DSO:

Total Acceleration = $\sqrt[2]{((\text{Acceleration x})^2 + (\text{Acceleration y})^{)\,2} + \ (\text{Acceleration z})^2)}$

When manipulated smoothly, the total acceleration's magnitude should always remain around the value of 1[g].

It would also be possible to integrate the acceleration twice and compute the position of the sensor in all directions.

Conclusions

The development, monitoring or maintenance of complex numerical systems is eased by using the right tools. A digital oscilloscope of the latest generation allows the observation of various signals at different points of the numerical processing chain. The simultaneous observation and correlation of analog, digital and encoded data allows rapid analysis of the defect and shortens the development cycles.

Thanks

Many thanks to F. Vannel and C. Abegg at the LSN for the opportunity to capture live signals on the system described here and the insights and discussions, as well as U. Schroffenegger and G. Ritter at LeCroy for the collaboration, insights and exchanges on the ProtoBusMAG module embedded in the LeCroy oscilloscopes.

The author

Roland Gamper is Senior Software Engineer and was employed 22 years by LeCroy in Geneva. He currently undertakes contract developments, in particular in the field of Protocol analysis, through Lahniss Ltd. He also teaches microcontroller programming at Geneva’s University of Applied Sciences.