

# Automated trace analysis for testing of CANopen devices

Andrew Ayre, Embedded Systems Academy, Inc.

**When it comes to testing of CANopen devices, one of the tests often conducted is the test of a device in a “golden system”. This is an integration test where the device under test is added to a known, good system. Unfortunately these tests are often conducted without any in-depth analysis – main objective is to see if the device works or does not work. This paper summarizes the functionality of an automated post analysis tool for trace recordings. Specific filtering allows for the generation of an “event log” that records only the main, system relevant events, warnings, errors and timings. It monitors the reliability of cycle times as well as correctness of LSS and SDO sequences. The reported results help to quickly pinpoint potential problems especially in dynamic CANopen systems (where some node IDs might be assigned dynamically).**

## Background

CANopen-based networks [1] are increasing in their sophistication and therefore complexity. Device profiles such as CiA447 Car Add-on Devices [2] add new challenges for network testing, debugging and confirmation of correctness.

Many of the issues in this paper relate to any CANopen network but CiA447 will be used as a real-world illustration as it uses many of the features that make testing and debugging more complex.

CiA447 is designed to allow plug and play of modules onto specialized passenger vehicles, such as radios, roof bars, taximeters, etc. A requirement is for a workshop to be able to swap out a faulty module without having to perform a systems integrator role. To facilitate this Layer Setting Services (LSS) are used for node ID assignment. Another requirement of the application profile is to conserve battery power when the vehicle is not in use. This is achieved through a wake-up and sleep power management protocol. Complexity is increased due to each module being responsible for going to sleep if the network is faulty or the manager is missing.

A roof bar needs to know which node is a roof bar controller, and therefore allowed to send commands. This necessitates every node “scanning” all other nodes on bootup to determine minimal identity and functionality information.

Taken together, these pieces of functionality combine to create dynamic networks where operation is only successful if multiple nodes communicate together at a high level and within specific time constraints.

If we look at how problems on these types of networks manifest themselves, we can see a variety of symptoms.

The network not going to sleep could be caused by a faulty manager, a talkative node or a node that failed to implement the sleep protocol properly, amongst others.

If a node sleeps too soon then it could wake up again and in turn stop other nodes from sleeping. The result is that a simple timing issue causes generation of many unexpected messages. Communication time outs could be caused by one node.

Failure of one node to scan another node could result in a breakdown of functionality – pressing a button on one node does not result in a light turning on at another node. If that light is a blue emergency light then this type of failure becomes critical.

The scanning aspect of a network can result in bursts of Service Data Object (SDO) messages during network startup, possibly at the same time as LSS is taking place. This is a prime opportunity for messages to be delayed and timeouts to occur, resulting in robust nodes deciding to autonomously retry operations or give up and enter an error state. In some standards the maximum message rate may be defined for types of

node, which is difficult to test for. Analysis of a log file can determine the actually maximum message rate.

The key aspect of testing and debugging such a system is that a failure in one area can lead to knock-on effects causing one or more failures in other areas, creating confusion as to the true cause of the problem.

**Basic spreadsheet analysis**

Applications that can record CAN messages on the bus and export to comma-separated value (CSV) files are common and easily obtainable. For a CANopen network it is highly beneficial to also have some level of message interpretation, allowing identification of the various types of messages.

Message Type	Node
Bootup	[0x06 (6)] <unknown>
Emergency	[0x06 (6)] <unknown>
CiA447 SDO Initiate Upload Request	C:0x01 S:0x03 - Client: [0
CiA447 SDO Upload Response	C:0x01 S:0x03 - Client: [0
CiA447 SDO Initiate Upload Request	C:0x01 S:0x06 - Client: [0
CiA447 SDO Upload Response	C:0x01 S:0x06 - Client: [0

Figure 1: Messages in a spreadsheet

Once in a spreadsheet it is relatively easy to find specific messages by searching for keywords such as “emergency” or “heartbeat”.

Provided timing information is including in the export, the relative time between specific messages can be calculated by finding the two messages of interest then subtracting one timestamp from the other.

However this approach is time consuming and error prone. With many nodes on a network there can be many heartbeat messages for example, and missing one may result in an incorrect analysis of the timing behavior. Calculating the time between messages in a spreadsheet is a multi-step process.

Spreadsheets typically have a limit on the number of rows that can be displayed. For Excel it is just over one million rows [3]. While at first glance this may appear perfectly adequate in reality it is a significant limitation for complex networks.

At 1 Mbps a message can take from 47 μs (no data, no stuff bits) to 130 μs (eight bytes,

maximum stuff bits) to be transmitted. For the worst case 100 % bus load:

$$1\ 048\ 576 \times 47\ \mu\text{s} = 49,28\ \text{seconds}$$

$$1\ 048\ 576 \times 130\ \mu\text{s} = 136,31\ \text{seconds}$$

Clearly this limitation is not suitable for analysis of log files generated over a period of hours or days, often a requirement for capturing the messages around infrequent problems.

When the number of messages becomes large, even while still fitting into a spreadsheet, manual analysis becomes increasing difficult and the opportunity for mistakes rises.

Consider the situation where LSS is used. Typically for multiple nodes the LSS cycle continues for node B while node A is booting and sending out messages. This results in LSS messages appearing mixed in with other unrelated messages. A spreadsheet can only show all messages in the order they appeared on the bus, and time consuming manipulation of the data is required to separate out LSS so it can be analyzed in isolation.

**What can be analyzed?**

Before considering an automated approach we first need to understand what we can analyze that would meet the needs of at least 90 % of applications.

For any application profile that uses sleep and wakeup being able to automatically detect these message is useful. They will show the start and ends of communication periods and therefore aid in identifying nodes that do not follow the protocol. The protocol consists of a small number of individual messages and therefore are easy to find.

A similar situation exists for bootup and emergency messages. In particular emergency messages can indicate a node has a problem, although this assumes the firmware of the node is operating correctly. Various timing information can be collected to highlight delayed and missing messages. The time between the transmissions of heartbeats from each node is easily determined. The time a node takes to respond to SDO requests can also be collected, which helps to highlight unusually long access times. The time between

Periodic Process Data Object (PDO) message transmissions from each node and verification of the inhibit times used can be collected.

Total TPDOs	9120
Total RPDOs	0
PDO 0x180 Repetition Times	Min: 98.771, Max: 104.098
PDO 0x181 Repetition Times	Min: 98.964, Max: 102.991

Figure 2: PDO analysis data

Moving to the next level more sophisticated analysis can take place which involves looking at the contents of the messages and understanding what they mean. This includes identifying an entire power cycle, from the moment the manager wakes the network up to the moment all nodes go to sleep. In a single log file there may be many such power cycles and identifying the beginning and end is useful.


	0:11:09:54.8736950	0x691	CiA447 Wake-Up...
--	--------------------	-------	-------------------

Figure 3: Wakeup message automatically identified in trace log using a power symbol

Entire LSS sequences can be monitored and the identities of each node found can be listed. Such information helps with systems integration, ensuring only the expected nodes are present. Out of sequence, e.g. unexpected messages, can be detected and flagged as an error. An example of a missing message that can be detected is the failure of an LSS slave to respond to assignment of a node identifier.

SDO segmented and block transfers can be monitored for overall transfer time, aborts, and total number of transfers. Unexpected and out-of-order messages can also be detected.

Numerical analysis is possible, collecting a variety of information on the characteristics of the network. This includes minimum, maximum and average times for periodic transmission and responses.

A running count can be maintained for total messages and for each type of message. Such a count allows easy detection of unknown/unidentified messages and also indicates which types of message use up the most bandwidth.

Once message counts and timing information is known the results can be compared against pre-defined warning and error levels. If any value exceeds one of these levels then the user can be notified.

Global analysis examples:

- Last time a message was seen, by type
- Total counts of message types
- Maximum message rate seen
- Longest message burst length
- Number of active nodes
- Minimum and maximum PDO periodic transmission times

Node-specific analysis examples:

- Total number of messages transmitted, by type
- Total number of times the timing values exceeded pre-defined warning and error levels
- Longest message burst and maximum message rate
- Number of SDO requests, responses and aborts.
- Largest SDO block transfer size
- SDO minimum and maximum response times
- Average PDO transmission rate
- Average time from reset to bootup
- Minimum and maximum heartbeat periodic transmission times
- Number of heartbeat losses
- Identity information

### Event detection

Along with generating numerical information about the operation of a network an automated system can apply intelligence to generate a list of interesting events. A user is then able to check the events and ensure there are no anomalies. This approach can be taken a step further and allow the list of events to be cross-referenced with the message log, allowing the context of the event to be determined.

	0:11:09:56.3742290	0x4E1	CiA447 SDO Up...
---	--------------------	-------	------------------

Figure 4: Trace log showing message that generated an event automatically indicated with an event symbol

Looking at the messages transmitted just before an event provides a convenient way to assess if this is a primary failure or a symptom of an earlier failure.

Events can be categorized according to interest level. Example categories could include standard, important, unexpected and error. A user would be able to use standard events for context, and which primarily serve to provide a “running commentary” on the analysis of the log file.

Type	Details
Unexpected	Not a system start
Unexpected	Unexpected PDO node 1
Important	Gateway wake up
Unexpected	Unexpected PDO node 10
Important	NMTmsg to 15:RESETap
Error	Emergency node 1 : EMCY 8130h-00h-0Fh 00h 00h 00h
Error	Emergency node 2 : EMCY 8130h-00h-0Fh 00h 00h 00h
Unexpected	Diag HB loss node 15

Figure 5: Example analysis events

For a system that uses sleep and wakeup the first expected message would be a wakeup message. If something else appears first then an unexpected system start event is generated.

An unexpected PDO event would indicate the transmission of a PDO before the node generating the PDO has booted and entered operational state. This could indicate a missing bootup message or a missing heartbeat message.

Examples of standard events:

- Time between NMT reset and node bootup
- Emergency reset message transmitted
- LSS setting a node ID or bit timing or activating a node
- First PDO from a node transmitted

Examples of important events:

- Pre-defined warning level reached
- Node bootup detected
- Missing emergency reset message
- NMT messages

Examples of unexpected events:

- Missing expected startup of network
- Heartbeat loss

- Failure to follow delays in wake/sleep protocol
- PDO transmitted by a node that is not known to be operational
- Invalid values for known object dictionary entries

Examples of error events:

- Message transmitted when network is sleeping
- Pre-defined error level reached
- Incorrect message length
- Emergency message transmitted
- Invalid NMT, LSS and SDO commands
- LSS and SDO protocol errors

### Implementation

There are two main use-cases for an automated analysis system. It can be used on-site by a systems integrator or workshop for example with a hand-held unit using “live” data, or it could be used with previously captured data in a PC application.

On-site with a hand-held unit provides instant results for a smaller data set size. The user can connect to the CAN bus, log messages, then instantly see the results of analysis include the events and numerical data. The sequence can then be repeated. For analyzing the network in a vehicle the hand-held unit can be left connected while the vehicle is operated, then retrieved and the results examined.

A PC application provides the opportunity for larger data sets to be examined with more detail provided. Log files with multi-millions of messages can be handled and results of the analysis can be exported for inclusion in reports.

Figure three shows a possible way of implementing such a dual-use system.

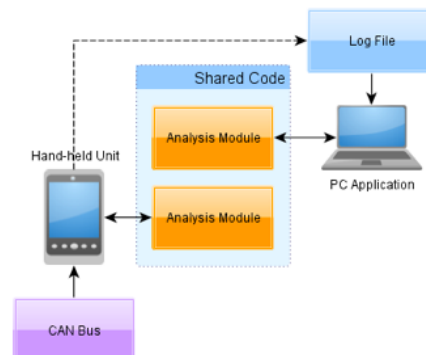


Figure 6: System architecture



The hand-held unit has firmware running on a microcontroller with CAN interface. During live monitoring of the network the messages are fed one at a time to the analysis module, which is part of the firmware. The analysis results continually update in real time.

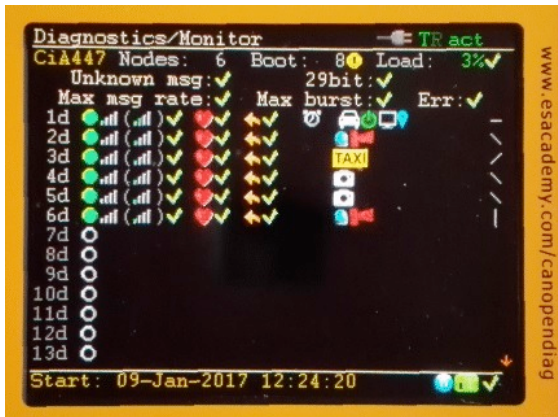


Figure 7: Analysis events on hand-held unit

The figure shows an overview of the current and maximum message rate, organized by node identifier. Also shown is the heartbeat status, SDO response times and identification details.

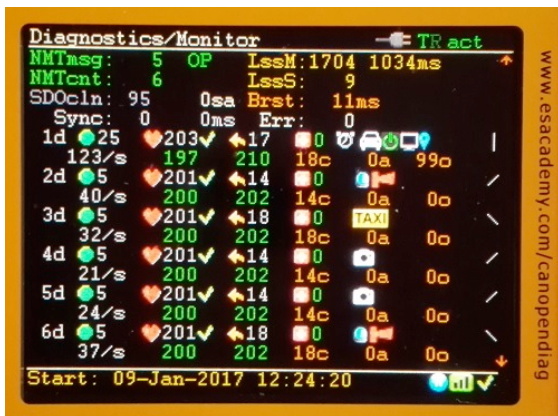


Figure 8: Detailed analysis on hand-held unit

The detailed analysis mode shows additional information, such as the range of heartbeat producer times and the number of SDO requests, aborts and sleep objections.

The PC application receives a pre-recorded log file, which could be in a variety of common file formats. The messages from the log are fed one at a time to the analysis module and the results are then presented to the user. Key to this system is the commonality employed. The analysis module is identical for both systems, reusing code and reducing

implementation time. It also guarantees the same analysis results for both systems, given the same input data.

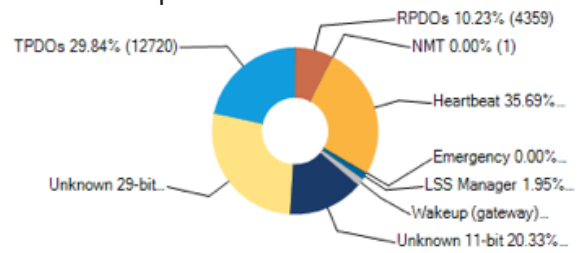


Figure 9: PC pie chart of message types

Both implementations allow the configuration of the overall system before analysis is started. For example the device profile being used is important to know in advance. In CiA447 sleep and wakeup cycles might need to be analyzed, however in CiA301 those same messages would be an error as they are not defined. Acceptable timing requirements can change from application to application. For one application an SDO response time of 100ms may be acceptable, but for another application and SDO response time of 30ms may be a hard requirement.

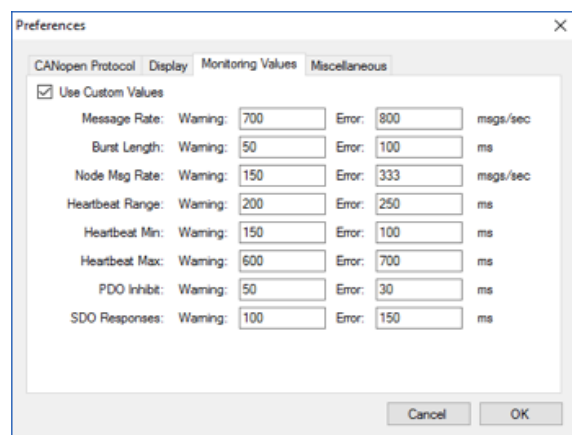


Figure 10: Example warning and error levels

The figure above shows an example of a range of warning and error levels that can be customized and applied to the analysis.

**Case studies**

11 099 messages were logged from a vehicle CANopen network [4]. On analysis the following errors were shown in the event list.

Important	NMTmsg to 2:RESETapp
Error	Emergency node 1 : EMCY 8130h-00h-02h 00h 00h 00h 00h
Error	Last SDO seq. incomplete

Figure 11: Events from analysis

The emergency event log entry is directly associated with a CAN message in the trace log. We examined that message to learn more.



Figure 12: Emergency message detail

This told us that node one (which was the manager in this network) had detected a heartbeat loss.

Using the trace log timestamps to scroll backwards more than 250 ms we confirmed there was no heartbeat transmission from the missing node. In this case the 250 ms period consisted of 23 messages. The use of the event log allowed us to quickly narrow down a problem area from over 11 000 messages to just 23. A significant time saver.

The second error shown states “Last SDO seq. incomplete”. We viewed the associated CAN message in the trace log for clues.

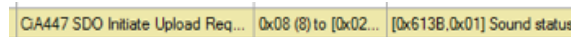


Figure 13: SDO request message detail

This showed an SDO request from node eight to node two. The error indicated that a previous SDO request was not completed at that point in time. Scrolling back five messages in the trace log showed the previous SDO message.

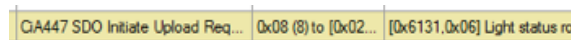


Figure 14: SDO request message detail

Clearly node eight already tried to access a different entry on node two. No response was received. Crucially node eight did not transmit an abort message before continuing. Looking at the timestamps we saw that 50 ms passed between the two SDO requests. We were able to easily identify a problem with both nodes by looking at just six messages in a large log file.

From a different vehicle CANopen log file [5] we viewed numerical information. For node one a graph made it clear the performance of heartbeat generation.

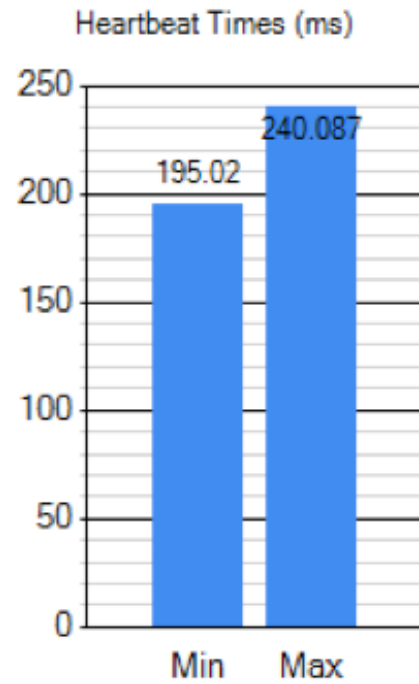


Figure 15: Heartbeat producer times (measured)

For CiA447 the required heartbeat producer time is 200 ms. We could see that this node was performing outside of the requirements and therefore would need further investigation.

A similar graph showed the performance of the SDO server on the same node.

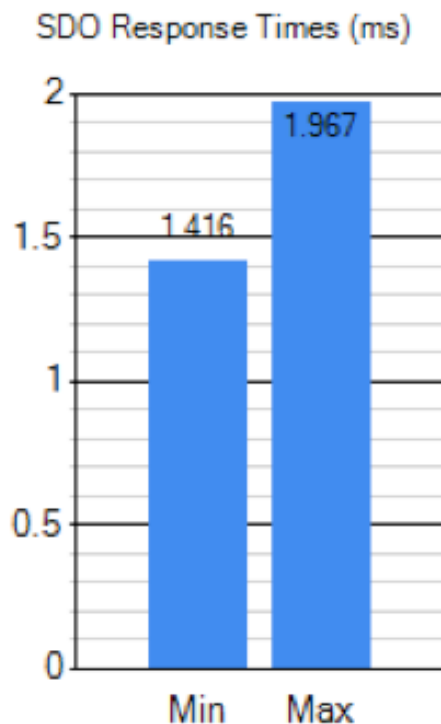


Figure 16: SDO response times (measured)

A large number for the maximum response time would indicate an issue, perhaps high bus load or a firmware functionality problem.

### Conclusion

For complex, dynamic networks where there is a large number of messages, node ID assignment and timing inter-dependencies automated analysis of network messages is a useful tool. A suitable application can quickly sort through and provide the developer and systems integrator with a range of information to assist in identifying problems and confirming correct network operation.

Less time spent on analyzing a network and locating problems is more time that can be spent on other activities.

### References

- [1] CiA DS 301, CANopen application layer and communication profile
- [2] CiA DSP 447, CANopen application profile for special-purpose car add-on devices
- [3] <https://support.office.com/en-us/article/Excel-specifications-and-limits-1672b34d-7043-467e-8e27-269d656771c3>
- [4] 21\_Mar\_2016\_10\_44\_58\_redacted.btr
- [5] Test1\_1st\_start.csv

---

Andrew Ayre  
Embedded Systems Academy, Inc.  
1250 Oakmead Parkway, Suite 210  
US-94085 Sunnyvale, CA  
Tel.: +1-877-812-6393  
Fax: +1-877-812-6382  
aayre@esacademy.com  
www.esacademy.com