Tractor Hacking

TractorHacking.github.io

Tim Letz ■ Andrew McGuan ■ AJ Fite

Project Description

John Deere farm equipment is used by farmers worldwide. Unfortunately the company is overly restrictive on the maintenance they allow the machines' owners to carry out. The company has imposed software checks that require a mechanic from a John Deere specific dealer, using equipment made available only to these dealers, to diagnose and fix errors. The errors have the potential to shut down the entire machine, even if it is isolated to a small or unimportant part of the tractor. If an owner tries to replace a faulty part without John Deere Software, the tractor can reject the part until a technician verifies the part. This is extremely limiting for the equipment owners, because it increases the downtime of the machinery, and transport and repair costs can be expensive.

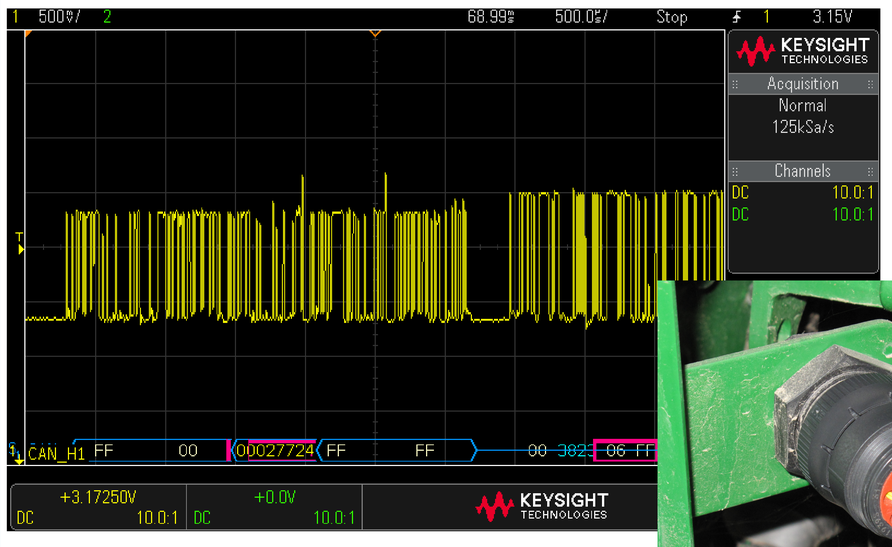
Objectives

This group intends to learn more about the computer architecture of John Deere tractors in an attempt to avoid their software restrictions. Our main objectives are:

- Obtain an engine control unit (ECU), which acts as the base of all communications on a tractor, learn about its architecture, and have it running in a benchtop setup
- Gain access to a live John Deere tractor and tap into its communication lines, to learn about the type of messages being sent during operation and periods of error
- Document our process on a publicly available webpage, so that others can pick up our work and continue where we left off

Design

The tractor used for live tests is a John Deere 5055E model, which is a fairly simple open-cab tractor. As a result, the computer architecture inside it is not incredibly complex. The tractor has a CAN port to which we connected our MSO-X 2012A Keysight oscilloscope. This scope is able to identify a message sent in the CAN format, break it up into its ID and data components, and save ten seconds of this data to a flashdrive.



Technical Detail

John Deere equipment conforms to a CAN protocol set by the Society of Automotive Engineers (SAE) named J1939. This is a standard that specifies the contents of the CAN message's ID, as well as what type of data is sent under specific ID numbers.

CAN EXTENDED FRAME FORMAT	SOF	IDENTIFIER 11 BITS											SRR	IDE	IDENTIFIER EXTENSION 18 BITS																RTR	...
J1939 FRAME FORMAT	SOF	PRIORITY			EDP	DP	PDU FORMAT (PF) 6 BITS (MSB)						SRR	IDE	PF (CONT.)		PDU SPECIFIC (PS) (DESTINATION ADDRESS, GROUP EXT. OR PROPRIETARY)								SOURCE ADDRESS								RTR	...
		3	2	1			8	7	6	5	4	3			2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	
J1939 FRAME BIT POSITION	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
CAN 29 BIT ID POSITION		28	27	26	25	24	23	22	21	20	19	18			17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

The above image is a breakdown of the 29-bit CAN ID sent with each message. The most important portion of this message is bits 8 through 23, which combine to form the parameter group number (PGN). J1939 specifies what data should be sent under each specific PGN. Shown below is a breakdown of the data sent under the PGN 0xF004.

Name	Size	Byte Offset
Engine Torque Mode	4b	1.1
Actual Engine - Percent Torque (Fractional)	4b	1.5
Driver's Demand Engine - Percent Torque	1B	2
Actual Engine - Percent Torque	1B	3
Engine Speed	2B	4-5
Source Address of Controlling Device for Engine Control	1B	6
Engine Starter Mode	4b	7.1
Engine Demand – Percent Torque	1B	8

J1939 specifies what data is being sent at each location in the CAN data region. This includes a description, the data size, and the byte offset in the data packet. This information is incredibly helpful in decoding the data gathered from the tractor, because it allowed us to look for specific IDs and observe the numbers changing at a specific point in the packet.

Data Analysis

The data we captured from the scope was stored in spreadsheets, in CAN format - further parsing and processing would have to be done on our own end - particularly, parsing these packets as according to the J1939 standard. To do so, we created a script that would read out data from these spreadsheets, and organize the data by various criteria. Through this, we were able to isolate which packets were John Deere-specific, and establish where to investigate further.

```
 0 (0x0000, 0, Name: Torque/Speed Control 1) (933 entries across all checked files)
 1 (0x0600, 1536, Name: CANopen Application Message #2/1) (891 entries across all checked files)
 2 (0x0c00, 3072, Name: Electronic Engine Controller 16) (286 entries across all checked files)
 3 (0x0e00, 3584, Name: Safety Header Message) (34 entries across all checked files)
 4 (0xea00, 59904, Name: Request) (37 entries across all checked files)
 5 (0xeb00, 60160, Name: Transport Protocol - Data Transfer) (104 entries across all checked files)
 6 (0xec00, 60416, Name: Transport Protocol - Connection Mgmt) (27 entries across all checked files)
 7 (0xee00, 60928, Name: Address Claimed) (55 entries across all checked files)
 8 (0xef00, 61184, Name: Proprietary A) (18057 entries across all checked files)
 9 (0xf003, 61443, Name: Electronic Engine Controller 2) (3491 entries across all checked files)
10 (0xf004, 61444, Name: Electronic Engine Controller 1) (4038 entries across all checked files)
11 (0xf005, 61445, Name: Electronic Transmission Controller 2) (703 entries across all checked files)
12 (0xf006, 61446, Name: Electronic Axle Controller 1) (137 entries across all checked files)
13 (0xfd7c, 64892, Name: Diesel Particulate Filter Control 1) (62 entries across all checked files)
14 (0xfdb3, 64947, Name: Aftertreatment 1 Outlet Gas 2) (143 entries across all checked files)
15 (0xfddf, 64991, Name: Front Wheel Drive Status) (134 entries across all checked files)
16 (0xfe43, 65091, Name: Primary or Rear Power Take off Output Shaft) (709 entries across all checked files)
17 (0xfe44, 65092, Name: Secondary or Front Power Take off Output Shaft) (700 entries across all checked files)
18 (0xfe49, 65097, Name: Ground-based Speed and Distance) (662 entries across all checked files)
19 (0xfe68, 65128, Name: Vehicle Fluids) (70 entries across all checked files)
20 (0xfe69, 65129, Name: Engine Temperature 3) (75 entries across all checked files)
21 (0xfeca, 65226, Name: Active Diagnostic Trouble Codes) (27 entries across all checked files)
22 (0xfed5, 65237, Name: Alternator Information) (84 entries across all checked files)
23 (0xfedf, 65247, Name: Electronic Engine Controller 3) (340 entries across all checked files)
24 (0xfee5, 65253, Name: Engine Hours, Revolutions) (15 entries across all checked files)
25 (0xfee7, 65255, Name: Vehicle Hours) (8 entries across all checked files)
26 (0xfeee, 65262, Name: Engine Temperature 1) (81 entries across all checked files)
```

Next Steps

The scope of our project has gotten a lot more concrete from the exploratory phase that our project started out in, and has definitely exceeded the time budget we're allotted for a single capstone. iFixit still plans on pursuing this topic in future years, and so our existing groundwork would be a great boon to future teams that will pick up this project and expand on our work. Much of our time was spent in research, looking up how to approach our task. With this already out of the way, groups in the future will be able to progress in a more straightforward fashion with the protocols in use highlighted and the equipment needed for analysis already at hand. On our website, we describe both packets we recognize from our captures, and our references used to define our analyses. We will hand our existing work off to iFixit for archival and future use.