How To Use AGL CAN Signal

05/31/2017
Yuichi Kusakabe
SS Engineering Group
Fujitsu TEN LIMITED
About Myself

- Yuichi Kusakabe (Fujitsu TEN LIMITED)
- Software Engineer of IVI about 10 years (for 16-bit and 32-bit architecture)
- Linux Software Engineer (2011–2013)
- Linux Software Lead Engineer (2013–Now)
- BSP Porting/Customizing
- Supporting for in-house software developers
- AGL (Automotive Grade Linux) Advisory Board member
Agenda

- What’s CAR CAN signal
- Standard Linux CAN IF & OSS CAN Tool
- How to use CAN signal to AGL
- Demonstration & Results
- Conclusion
What’s CAR CAN Signal
What's CAR CAN Signal

Standard CAN Signal is Low Speed (500kbps), But High frequency (\(\text{**} \text{us}\)).

- Standard CAN Signal format (11bit).
- Data line: D+/D−/GND (want)
- Baud rate: 500kbps
- CAN ID: 11bit (0x000~0x7FF)
- Data size: 0~8byte
- CAN Bus load: 20~75%

https://ja.wikipedia.org/wiki/Controller_Area_Network
Standard Linux CAN IF & OSS CAN Tool
Standard Linux CAN IF (SocketCAN)

Linux kernel all ready CAN IF with Socket CAN

SocketCAN

From Wikipedia, the free encyclopedia

SocketCAN is a set of open source CAN drivers and a networking stack contributed by Volkswagen Research to the Linux kernel. Formerly known as Low Level CAN Framework (LLCF).

Traditional CAN drivers for Linux are based on the model of character devices. Typically they only allow sending to and receiving from the CAN controller. Conventional implementations of this class of device driver only allow a single process to access the device, which means that all other processes are blocked in the meantime. In addition, these drivers typically all differ slightly in the interface presented to the application, stifling portability. The SocketCAN concept on the other hand uses the model of network devices, which allows multiple applications to access one CAN device simultaneously. Also, a single application is able to access multiple CAN networks in parallel.

The SocketCAN concept extends the Berkeley sockets API in Linux by introducing a new protocol family, PF_CAN, that coexists with other protocol families like PF_INET for the Internet Protocol. The communication with the CAN bus is therefore done analogously to the use of the Internet Protocol via sockets. Fundamental components of SocketCAN are the network device drivers for different CAN controllers and the implementation of the CAN protocol family. The protocol family, PF_CAN, provides the structures to enable different protocols on the bus: Raw sockets for direct CAN communication and transport protocols for point-to-point connections. Moreover the broadcast manager which is part of the CAN protocol family provides functions e.g. for sending CAN messages periodically or realize complex message filters.

Patches about CAN were added in the 2.6.25 Linux kernel. Meanwhile some controller drivers were added and work is going on to add drivers for a variety of controllers.

Standard Linux CAN IF(CAN Driver)

Linux kernel all ready CAN IF with Socket CAN

Readme file for the Controller Area Network Protocol Family (aka SocketCAN)

This file contains

1 Overview / What is SocketCAN
2 Motivation / Why using the socket API
3 SocketCAN concept
   3.1 receive lists
   3.2 local loopback of sent frames
   3.3 network problem notifications
4 How to use SocketCAN
   4.1 RAW protocol sockets with can_filters (SOCK_RAW)
      4.1.1 RAW socket option CAN_RAW_FILTER
      4.1.2 RAW socket option CAN_RAW_ERR_FILTER
      4.1.3 RAW socket option CAN_RAW_LOOPBACK
      4.1.4 RAW socket option CAN_RAW_RECV_OWN_MGS
      4.1.5 RAW socket option CAN_RAW_FD_FRAMES
      4.1.6 RAW socket option CAN_RAW_JOIN_FILTERS
      4.1.7 RAW socket returned message flags
   4.2 Broadcast Manager protocol sockets (SOCK_DGRAM)
      4.2.1 Broadcast Manager operations
      4.2.2 Broadcast Manager message flags
      4.2.3 Broadcast Manager transmission timers
      4.2.4 Broadcast Manager message sequence transmission
      4.2.5 Broadcast Manager receive filter timers
      4.2.6 Broadcast Manager multiplex message receive filter
      4.2.7 Broadcast Manager CAN FD support
   4.3 connected transport protocols (SOCK_SEQPACKET)
   4.4 unconnected transport protocols (SOCK_DGRAM)

5 SocketCAN core module
   5.1 can.ko module params
   5.2 procs content
   5.3 writing own CAN protocol modules

https://www.kernel.org/doc/Documentation/networking/can.txt

CONFIG_CAN=y
CONFIG_CAN_RAW=y
CONFIG_CAN_BCM=y
CONFIG_CAN_GW=y
CONFIG_CAN_RCAR=y
OSS CAN Tool (Powerful software)

can-utils easy to debug CAN Signal (read/write/play)

https://github.com/linux-can/can-utils
**OSS CAN Tool (Powerful software)**

### CAN data send (cansend)

- ID=333 (11bit), DATA=33 send=can0
  - `# cansend can0 333#33`

- ID=00004444 (24bit), DATA=44 send=can0
  - `# cansend can0 00004444#44`

### CAN data recv (candump)

- `recv=can`
- `# candump can0 -ta`
- `root@porter:~# candump can0 -ta`
  - (1478869757.430017) can0 344 [8] FF EE 00 00 00 00 EE AA
  - (1478869757.431290) can0 226 [8] E4 00 00 EE 00 EE EE 00

- `recv=all`
- `# candump any -ta`
How to use CAN signal to AGL
This time AGL provide AMB, but AGL remake new CAN Signal handing FW.

https://github.com/otcshare/automotive-message-broker

This is OLD software
Low level CAN service

Low level CAN service made to decode and write on CAN bus

https://gerrit.automotivelinux.org/gerrit/gitweb?p=apps/low-level-can-service.git;a=summary
CAN key technology

- CAN ID Filtering and CAN DATA Thinning out is very important

- CAN signal filtering (setting SocketCAN)
  - CAN ID xxx → xx

- CAN signal thinning out
  - CAN cycle xx ms → xxx ms

- CAN data convert AGL Standard
AGL CAN Software stack

AGL support easy to use CAN data

AGL Simple CAN Simulator

Sample CAN Apps

Low level CAN service (afb)

Sample CAN Apps

can-utils

slcan0

can0

vcan0

/dev/input/js0

R-CAR M3

USB-HID

USB-Serial

CAN
The OpenXC Platform

OpenXC™ is a combination of open source hardware and software that lets you extend your vehicle with custom applications and pluggable modules. It uses standard, well-known tools to open up a wealth of data from the vehicle to developers, even beyond OBD-II.

What is OpenXC™?

OpenXC™ is an open source, data-focused API for your car. By installing a small hardware module, the vehicle data becomes accessible to Android or other desktop applications using the OpenXC library.

Unlocking Rich Vehicle Data

OpenXC™ allows consumer devices, such as smart phones, to access data from any vehicle. Using OpenXC™, you can monitor many of the sensors on a vehicle, enabling new and innovative vehicle-centric applications. Some data is required by law and more can be unlocked with support from an automaker - or a little reverse engineering effort!

http://openxcplatform.com/
Official Signals

These signal names are a part of the OpenXC specification, although some manufacturers may support custom message names.

- steering_wheel_angle
  - numerical, -600 to +600 degrees
  - 10Hz
- torque_at_transmission
  - numerical, -500 to 1500 Nm
  - 10Hz
- engine_speed
  - numerical, 0 to 16382 RPM
  - 10Hz
- vehicle_speed
  - numerical, 0 to 655 km/h (this will be positive even if going in reverse as it's not a velocity, although you can use the gear status to figure out direction)
  - 10Hz
- accelerator_pedal_position
  - percentage
  - 10Hz
- parking_brake_status
  - boolean, (true == brake engaged)
  - 1Hz, but sent immediately on change
Public CAN protocol (FMS)

FMS-Standard description

Version 03


© HDEI / BCEI Working Group

### Public CAN protocol (FMS)

<table>
<thead>
<tr>
<th>Page</th>
<th>PGN</th>
<th>SPN</th>
<th>e.g. mileage, fuel consumption</th>
<th>Mandatory</th>
<th>rep. rate</th>
<th>remarks / comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>65257</td>
<td>250</td>
<td>Truck AND Bus Section</td>
<td>Truck only</td>
<td>1000</td>
<td>4 bytes, 0 to +2 105 540 607.5 L</td>
</tr>
<tr>
<td>8</td>
<td>65276</td>
<td>96</td>
<td>X (worldwide)</td>
<td>1 Byte</td>
<td>20</td>
<td>1 % Bit, -125 % offset</td>
</tr>
<tr>
<td>9</td>
<td>61444</td>
<td>110</td>
<td>engine speed</td>
<td>2 Byte</td>
<td>20</td>
<td>8031.875 rpm</td>
</tr>
<tr>
<td>10</td>
<td>65253</td>
<td>247</td>
<td>Truck AND Bus Section</td>
<td>X (worldwide)</td>
<td>1000</td>
<td>4 bytes, 0 to 219 554 060.75 h</td>
</tr>
<tr>
<td>11</td>
<td>65260</td>
<td>235</td>
<td>Vehicle identification number</td>
<td>X (worldwide)</td>
<td>10000</td>
<td>Will be sent every 10 sec</td>
</tr>
<tr>
<td>12</td>
<td>64977</td>
<td>2806</td>
<td>SW-version supported</td>
<td>X (worldwide)</td>
<td>10000</td>
<td>Indicator for SW version supported</td>
</tr>
<tr>
<td>13</td>
<td>64977</td>
<td>2804</td>
<td>Diagnostics supported</td>
<td>X (worldwide)</td>
<td>10000</td>
<td>indicator for diagnosis session support</td>
</tr>
<tr>
<td>14</td>
<td>64977</td>
<td>2803</td>
<td>Requests supported</td>
<td>X (worldwide)</td>
<td>10000</td>
<td>indicator for request support</td>
</tr>
<tr>
<td>15</td>
<td>64127</td>
<td>197</td>
<td>High resolution total vehicle distance</td>
<td>X (worldwide)</td>
<td>10000</td>
<td>4 bytes, 0 to 21 05 606 km, without TCO</td>
</tr>
<tr>
<td>16</td>
<td>65132</td>
<td>1611</td>
<td>Vehicle motion</td>
<td>X (EU)</td>
<td>20/60</td>
<td>Resolution may be not within the SAE values</td>
</tr>
<tr>
<td>17</td>
<td>65132</td>
<td>1613</td>
<td>driver 2 working state</td>
<td>X (EU)</td>
<td>20/60</td>
<td>rep. rate tach dependant</td>
</tr>
<tr>
<td>18</td>
<td>65132</td>
<td>1614</td>
<td>driver 1 working state</td>
<td>X (EU)</td>
<td>20/60</td>
<td>rep. rate tach dependant</td>
</tr>
<tr>
<td>19</td>
<td>65132</td>
<td>1615</td>
<td>Driver 1 time relat. states</td>
<td>X (EU)</td>
<td>20/60</td>
<td>rep. rate tach dependant</td>
</tr>
<tr>
<td>20</td>
<td>65132</td>
<td>1616</td>
<td>Driver 2 time relat. states</td>
<td>X (EU)</td>
<td>20/60</td>
<td>rep. rate tach dependant</td>
</tr>
<tr>
<td>21</td>
<td>65132</td>
<td>1617</td>
<td>Tachograph performance</td>
<td>X (EU)</td>
<td>20/60</td>
<td>rep. rate tach dependant</td>
</tr>
<tr>
<td>22</td>
<td>65132</td>
<td>1619</td>
<td>Direction indicator</td>
<td>X (EU)</td>
<td>20/60</td>
<td>rep. rate tach dependant</td>
</tr>
<tr>
<td>23</td>
<td>65132</td>
<td>1620</td>
<td>Tachograph performance</td>
<td>X (EU)</td>
<td>20/60</td>
<td>rep. rate tach dependant</td>
</tr>
<tr>
<td>24</td>
<td>65132</td>
<td>1621</td>
<td>Handling information</td>
<td>X (EU)</td>
<td>20/60</td>
<td>rep. rate tach dependant</td>
</tr>
<tr>
<td>25</td>
<td>65132</td>
<td>1622</td>
<td>System event</td>
<td>X (EU)</td>
<td>20/60</td>
<td>rep. rate tach dependant</td>
</tr>
<tr>
<td>26</td>
<td>65132</td>
<td>1623</td>
<td>Tachograph vehicle speed</td>
<td>X (EU)</td>
<td>20/60</td>
<td>rep. rate tach dependant</td>
</tr>
<tr>
<td>27</td>
<td>65262</td>
<td>110</td>
<td>Engine coolant temperature</td>
<td>X (worldwide)</td>
<td>1000</td>
<td>0.03125 °C / Bit gain</td>
</tr>
<tr>
<td>28</td>
<td>65269</td>
<td>171</td>
<td>Ambient Air Temperature</td>
<td>X (worldwide)</td>
<td>1000</td>
<td>-273 C offset</td>
</tr>
<tr>
<td>29</td>
<td>65131</td>
<td>1625/1626</td>
<td>Driver 1 / Driver 2 Identification</td>
<td>X (EU)</td>
<td>10000</td>
<td>If a driver ID is available the message is sent with a Broadcast Announce Message (BAM)</td>
</tr>
<tr>
<td>30</td>
<td>65266</td>
<td>183</td>
<td>Fuel rate</td>
<td>X (worldwide)</td>
<td>100</td>
<td>0.05 L/h per Bit, 0 to 3.0217.5 L/hr</td>
</tr>
<tr>
<td>31</td>
<td>65266</td>
<td>184</td>
<td>Instantaneous Fuel Economy</td>
<td>X (worldwide)</td>
<td>100</td>
<td>1/15 km/L per Bit, 0 to 125,5 km/L</td>
</tr>
<tr>
<td>32</td>
<td>65198</td>
<td>1087</td>
<td>Brake Air Pressure Circuit 1</td>
<td>X (worldwide)</td>
<td>100</td>
<td>1 L/km at 100 km/h</td>
</tr>
<tr>
<td>33</td>
<td>65198</td>
<td>1088</td>
<td>Service Brake Air Pressure Circuit 2</td>
<td>X (worldwide)</td>
<td>100</td>
<td>1.8 kPa/lit, 0 offset</td>
</tr>
<tr>
<td>34</td>
<td>64777</td>
<td>5054</td>
<td>High resolution total fuel used</td>
<td>X (worldwide)</td>
<td>10000</td>
<td>0.001141, 0 to 4,211,081.215 L</td>
</tr>
<tr>
<td>35</td>
<td>65110</td>
<td>1761</td>
<td>Aftertreatment 1 Diesel Exhaust Fluid Tank 1 Level</td>
<td>X (worldwide)</td>
<td>1000</td>
<td>0.4 % bit, 0 offset</td>
</tr>
</tbody>
</table>

56 CAN Data (include Bus)

Demonstration & Results
### AGL reference Hardware to Renesas R-CAR M3

<table>
<thead>
<tr>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-Car M3搭載</td>
</tr>
<tr>
<td>• ARM® Cortex-A57 (ARMv8) 1.7 GHz dual core, with NEON/VFPv4, L1 cache I/D 48K/32K, L2 cache 2MB</td>
</tr>
<tr>
<td>• ARM Cortex-A53 (ARMv8) 1.3 GHz quad core, with NEON/VFPv4, L1 cache I/D 32K/32K, L2 cache 512K</td>
</tr>
<tr>
<td>• memory controller for LPDDR4 2GB in 2 channels, each 32-bit wide</td>
</tr>
<tr>
<td>• three-dimensional graphics engines</td>
</tr>
<tr>
<td>• video processing units</td>
</tr>
<tr>
<td>• 3 channels display output</td>
</tr>
<tr>
<td>• 6 channels video input</td>
</tr>
<tr>
<td>• SD card host interface</td>
</tr>
<tr>
<td>• USB3.0 and USB2.0 interfaces</td>
</tr>
<tr>
<td>• CAN interfaces</td>
</tr>
<tr>
<td>• Ethernet AVB</td>
</tr>
<tr>
<td>• PCI Express Interfaces</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>メモリ</th>
</tr>
</thead>
<tbody>
<tr>
<td>• internal 384KBytes system RAM</td>
</tr>
<tr>
<td>• 64 MBytes HyperFlashTM (512 Mbits)</td>
</tr>
<tr>
<td>• 16MBytes QSPI flash(128 Mbits) 1 header QSPI module</td>
</tr>
<tr>
<td>• 8 GBytes eMMC</td>
</tr>
<tr>
<td>• microSD card slot</td>
</tr>
</tbody>
</table>

R-CarスタータキットがAutomotive Grade Linuxの標準リファレンスプラットフォームに採用、次世代コネクテッドカーのIVI開発を加速
～最新のUnified Code Base（UCB）3.0の64ビットソフトウェア環境に対応～

2017年05月24日
ルネサス エレクトロニクス株式会社

ルネサス エレクトロニクス株式会社（代表取締役社長兼CEO：呉 文精、以下ルネサス）は、このたびR-Carスタータキットが、Linuxベースの車載情報機器のオープンソースプロジェクトAutomotive Grade Linux（AGL）のソフトウェア開発用標準リファレンスプラットフォームの1つに採用されたことを発表しました。これにより、同プロジェクトが開発したソフトウェアを動かすハードウェア環境を容易に入手可能となり、コネクテッドカー時代に向けて、IVI（In-Vehicle Infotainment）用アプリケーションソフトウェアが支援されることが可能となります。

また、このたびR-Carスタータキットと組み合わせて使用するIVI開発用拡張ボード2種類（シマフジ電機社製）が7月より発売されます。拡張ボードのスタンダードモデルは、マルチディスプレイや各種ネットワーク用インタフェースを搭載、さらにアドバンスドモデルは、最大8チャンネルのカメラ入力や高速/大容量のストレージを拡張できるインタフェースを装備しています。

CANUSB easy connect CAN IF

CAN Tools

Home  News  Products  Projects  Downloads  Where to Buy?  About

CANUSB

General Information:
CANUSB is a very small dongle that plugs into any PC USB Port and gives an instant CAN connectivity. This means it can be treated by software as a standard COM Port (virtual serial RS232 port) with the FTDI USB drivers which eliminates the need for any extra drivers (DLL) or by installing a direct driver DLL (D2XX) together with our CANUSB DLL for faster communications and higher CAN bus loads. Sending and receiving can be done in standard ASCII format.

http://www.can232.com/?page_id=16
Copyright © 2017 FUJITSU TEN LIMITED. All rights reserved.
Target Hardware spec

CANUSB connected CAN IF simple Hardware

CAN Pin assignement:

CAN_GND / nc
CAN_H
nc
+ 12V / + 5V / nc
CAN_L
CAN_GND / nc
nc
nc

Pin assignement according to CiA recommendations DS102-1.
The CANUSB is powered from USB port, so no need to connect external power on pin 9. Use only CAN_L (pin2), CAN_H (Pin7) and CAN_GND (pin3).

The picture above shows how to connect the CANUSB (click here for a larger view). No external power is needed, the CANUSB uses 5VDC/100mA from USB.
Software setup

Add Kernel defconfig CAN driver and CANUSB

- CONFIG_CAN=y
- CONFIG_CAN_VCAN=y
- CONFIG_CAN_RCAR=y  \(\Leftarrow\) Renesas board only
- CONFIG_CAN_SLCAN=y
- CONFIG_USB_SERIAL=y
- CONFIG_USB_SERIAL_FTDI_SIO=y

Add rootfs “can-utils” and “iproute2”

- yocto local.conf
- IMAGE_INSTALL_append = “can-utils iproute2”

Setup CAN and CANUSB

**CAN0**
- ip link set can0 type can bitrate 500000
- ip link set can0 up

**CANUSB**
- slcand -o -s 6 -t hw /dev/ttyUSB*
- ip link set slcan0 up
Software setup

Start Low level CAN service (afb-daemon)

```bash
./afb-daemon --token=${AFB_CANIVI_TOKEN} --ldpaths=.
--port=${AFB_CANIVI_PORT}
--rootdir=.
```

Setting CAN data receive

```bash
./afb-client-demo ws://localhost:5555/api?token=3210

canivi subscribe { "event" : "*" } <- receive all data

canivi subscribe { "event" : "VehicleSpeed" } <- receive Vehicle Speed only
```
Demonstration
Performance measurement result

- Data used for measurement
  - logtime = 1122 sec,
  - CAN ID cnt = 129, datacnt = 1042673
  - can load ave= 19.04%, min = 18.82%, max = 37.89%
  - cyc ave[us] = 929, min = 8, max = 8565
- Support CAN ID 42, Thinning out time ** $\rightarrow$ 100ms

<table>
<thead>
<tr>
<th>AMB(d-bus)</th>
<th>Process name</th>
<th>CPU load(%)</th>
<th>AMB/AFB (vs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Process name</td>
<td>CPU load(%)</td>
<td>AMB/AFB (vs)</td>
</tr>
<tr>
<td>Service</td>
<td>ambd</td>
<td>16.97</td>
<td>1.34</td>
</tr>
<tr>
<td></td>
<td>dbus-daemon</td>
<td>6.33</td>
<td></td>
</tr>
<tr>
<td>Client(App)</td>
<td>AMBDBus_tp</td>
<td>4.26</td>
<td>0.82</td>
</tr>
<tr>
<td></td>
<td>afb-client-demo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAN Sim</td>
<td>canplayer</td>
<td>2.36</td>
<td>1.51</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>*</td>
<td>30.65</td>
<td>3.67</td>
</tr>
</tbody>
</table>
Next future

Support all in-vehicle communication (many CAN DATA and many Vehicle IF)

OEM DATA Center

AGL CAN Software stack

Sample CAN Apps

Sample CAN Apps

Low level CAN service (afb)

Original driver

can0

can1

OEM original interface

R-CAR M3

CAN0

CAN1

Copyright © 2017 FUJITSU TEN LIMITED. All rights reserved.
Linux Kernel all ready use to CAN

OSS CAN Tool “can utils” is good software

AGL support easy to use CAN data

Next step
- Define AGL public CAR CAN data format
  -> With the cockpit architecture team
- AGL simple CAN simulator provide
- Support all in-vehicle communication
Extra information
この記事は、Fujitsu Advent Calendar 2枚目の1日目の記事です。

はじめに

組み込みLinux評価ボードでCAN(車載通信)を使用する方法について調べてみました。
CANには色々な規格がありますが、今回は一般的な500kビット/秒の標準フォーマット(11bit)を使用してみました(CANの詳細はwikipedia等を参照して下さい)。

1.ターゲットボードの用意

手頃な価格でCANを使用出来る組み込み評価ボードが無いか調べてみましたが、SoCがCANに対応していても評価ボードにCAN IFが搭載されていないボードが大半です。今回はAutomotiveGradeLinuxの標準ブリファレンスボードに指定されているRenesas社のPorterボードを使用してみます。

ボードのCAN IFについては下記に詳しく説明が記載されています。

http://qiita.com/yuichi-kusakabe/items/e5b50aa3edb712bb6916
Thank you!!!

yuichi.kusakabe@jp.fujitsu.com