How To Use Linux CAN Signal To AGL
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
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 (***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_MSGS
      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-k0 module params
   5.2 procs content
   5.3 writing own CAN protocol modules

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

Copyright © 2016 FUJITSU TEN LIMITED. All rights reserved.
OSS CAN Tool (Powerful software)

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

https://github.com/linux-can/can-utils

Copyright © 2016 FUJITSU TEN LIMITED. All rights reserved.
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
  ```

https://github.com/linux-can/can-utils
How to use CAN signal to AGL
AMB (Automotive Message Broker)

This time AGL provide AMB, but AGL remake new CAN Signal handing FW.

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

AMB provide simple plugin only, default is not use SocketCAN

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

Copyright © 2016 FUJITSU TEN LIMITED. All rights reserved.
What’s CANRAWPLUGIN

CANRAWPLUGIN is Simple SocketCAN AMB Plugins

- CAN signal filtering (setting SocketCAN)
  - CAN ID xxx → xx
- CAN signal thinning out
  - CAN cycle xx ms → xxx ms
- CAN data convert AMB format
Demonstration & Results
Target Hardware spec

AGL reference Hardware to Renesas R-CAR M2 Porter board

- R-Car M2 SoC
  - ARM® Cortex-A15 Dual Core 1.5GHz
  - Multimedia Engine SH4A 780 MHz
  - GPU
    - PowerVR SGX544MP2 (3D)
    - Renesas graphics processor (2D)
- 2 GB DDR3 memory (dual channel)
- Two flash memory chips
  - 4 MB SPI
  - 64 MB SPI
- Debug Ethernet (100 Mbps)
- Storage connection
  - one SATA rev. 3.1 port
  - one SD card slot
  - one microSD card slot
- Analog Video In: ADV7180 Video Decoder
  - RCA jack
  - NTSC/PAL/SECAM autodetection
- Audio codec: AK4643EN
  - Line In 3.5 mm jack
  - LineOut 3.5 mm jack
- Two USB 2.0 ports
  - microUSB port supports host, device and OTG modes
- PCI Express x1 slot
- CAN transceiver

http://elinux.org/R-Car/Boards/Porter

32GB microSDHC

http://panasonic.jp/sd/p-db/RP-SMGB32GJK.html
Target Hardware spec

CAN simulator running to AGL reference Hardware to Renesas R-CAR M3

http://elinux.org/R-Car/Boards/

Copyright © 2016 FUJITSU TEN LIMITED. All rights reserved.
Target Hardware spec

R-CAR M3 not include CAN IF, CANUSB easy connect CANIF

CAN Tools

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
Target Hardware spec

CANUSB connected CAN IF simple Hardware

CAN Pin assignment:

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+12V/+5V/nc</td>
</tr>
<tr>
<td>2</td>
<td>CAN_L</td>
</tr>
<tr>
<td>3</td>
<td>CAN_GND/nc</td>
</tr>
<tr>
<td>4</td>
<td>nc</td>
</tr>
<tr>
<td>5</td>
<td>nc</td>
</tr>
<tr>
<td>6</td>
<td>CAN_GND/nc</td>
</tr>
<tr>
<td>7</td>
<td>CAN_H</td>
</tr>
<tr>
<td>8</td>
<td>nc</td>
</tr>
<tr>
<td>9</td>
<td>+12V/+5V/nc</td>
</tr>
</tbody>
</table>

Pin assignment 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.

http://www.can232.com/?page_id=16
Software setup

Add Kernel defconfig CAN driver and CANUSB

```
CONFIG_CAN=y
CONFIG_CAN_VCAN=y
CONFIG_CAN_RCAR=y  <- Renesas Porter 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
```
Demonstration

Porter board (running AGL CES DEMO), M3 (running CAN simulator) and CANUSB
## Results

- **CAN data detail**
  - Filtering CAN ID:124 → 40
  - Thinning out time: **ms → 100ms

<table>
<thead>
<tr>
<th>No</th>
<th>CAN Bus load</th>
<th>can id/sec</th>
<th>CPU load (AMB + d-bus)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0%</td>
<td>0</td>
<td>6.7%</td>
</tr>
<tr>
<td>2</td>
<td>29%</td>
<td>1,447</td>
<td>35.88%</td>
</tr>
<tr>
<td>3</td>
<td>43%</td>
<td>2,063</td>
<td>86.29%</td>
</tr>
<tr>
<td>4</td>
<td>50%</td>
<td>Unmeasurable -&gt; data lost</td>
<td>Unmeasurable -&gt; data lost</td>
</tr>
</tbody>
</table>

*AMB and d-bus is heavy, and small CAN data handing difficult*
Conclusion

- Linux Kernel all ready use to CAN
- OSS CAN Tool “can utils” is good software
- CAN Signal handing resource is difficult

Next step
- Define AGL public CAR CAN data format
- AGL standard CAN simulator
- New CAN handing FW support SocketCAN
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、今回のは一般的な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の標準ライセ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した。

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した。

http://qiita.com/yuichi-kusakabe/items/e5b50aa3edb712bb6916

Copyright © 2016 FUJITSU TEN LIMITED. All rights reserved.
Thank you!!!

yuichi.kusakabe@jp.fujitsu.com