7 November 2024

Installation and Utilization of ATLAS Modified SystecCAN Driver for USB-CANmodul Series

ATLAS collaboration

To be utilized with the USB-CANmodul series generation 3 and 4; and a AlmaLinux 9.3 machine with kernel 5.14.0-362.18.1.el9_3.x86_64

Summary

This document serves as a comprehensive guide for the installation and utilization of the ATLAS Modified SystecCAN Driver, specifically tailored for the USB-CANmodul Series Generation 3 and 4. It is compatible with AlmaLinux 9.3 machines running kernel 5.14.0-362.18.1.el9 3.x86_64. The guide details a step-by-step process starting from downloading the driver from the GitLab repository to compiling, installing, and verifying the installation. Additionally, it covers configuring CAN interfaces, testing CAN communication, and provides usage examples along with an automated testing script.

ATLAS Doc:		
EDMS Id:		
Version:	1.1	
Last modified:	`17:32 on 7 November 2024`	
Prepared by: S. Vergara C. Rodriguez C. Melo J Daza	**Checked by:**	**Approved by:**

Contents

List of Figures

List of Tables

1 Introduction

This document provides a comprehensive guide on the installation and utilization of the ATLAS Modified SystecCAN Driver for USB-CANmodul Series. This modified driver, designed for Systec USB-CAN hardware systems, ensures compatibility with AlmaLinux 9.3 (Kernel 5.14.0-362.18.1.el9_3.x86_64). The purpose of this document is to assist users in setting up the driver, understanding its functionalities, and troubleshooting common issues.

The ATLAS Modified SystecCAN Driver for USB-CANmodul Series is a tailored version of the Systec driver, adapted for specific kernel compatibility and enhanced with additional features for error handling and controller modes. It is essential for integrating USB-CANmodul hardware with systems running AlmaLinux, facilitating CAN communication for various applications in research and development environments.

2 Installation of ATLAS Modified SystecCAN Driver

This section provides detailed instructions on how to install the ATLAS Modified SystecCAN Driver for the USB-CANmodul Series.

2.1 Prerequisites

- AlmaLinux 9.3
- Kernel version 5.14.0-362.18.1.el9_3.x86_64
- USB-CANmodul hardware from Systec

Important Note (Running CAN Modules) 2-1

2.2 Running the CAN Modules

Before using the CAN interface, you need to load the necessary modules. Use the following commands to load the CAN modules:

```
sudo modprobe can
sudo modprobe can_raw
sudo modprobe can_dev
```

These commands will load the core CAN modules, the raw CAN protocol, and the CAN device drivers respectively.

2.3 Running CAN Modules Automatically at Boot

To ensure the CAN modules load automatically at boot, add them to the `/etc/modules-load.d/can.conf` file:

```
echo -e "can\ncan_raw\ncan_dev" | sudo tee /etc/modules-load.d/can.conf
```

This command creates a configuration file that lists the CAN modules to be loaded at startup.

2.3.1 Downloading the Driver

First, download the driver from the GitLab repository. Open a terminal and execute the following commands:

```
git clone
↪  https://gitlab.cern.ch/vergaraa/Adapted-systec_can-v1.0.7-Driver-for-ATLAS.git
cd Adapted-systec_can-v1.0.7-Driver-for-ATLAS
```

This will clone the repository to your local machine and change the directory to the newly downloaded repository.

2.3.2 Compiling the Driver

Next, within the folder that all the files are, compile the driver by running:

```
make all
```

This command will use the Makefile provided in the repository to build the driver.

2.3.3 Installing the Driver

To install the compiled driver, execute:

```
sudo make install
```

This command requires root privileges, hence the use of ‘sudo‘.

2.3.4 Loading the Kernel Module

After installation, load the kernel module using (run this in the root folder of the compiled driver):

```
sudo insmod systec_can.ko
```

This command will insert the ‘systec_can‘ module into the kernel.

2.3.5 Verifying the Installation

To verify that the driver is correctly installed and loaded, check the system messages:

```
dmesg | grep systec_can
```

You should see output related to the ‘systec_can‘ module, confirming its successful loading.

Important Note (Automatic Driver Loading) **2-2**

Please note that the driver module (`systec_can.ko`) must be loaded each time the system is rebooted. To automate this process, add the module name to your `/etc/modules` file:

```
echo "systec_can" | sudo tee -a /etc/modules
```

This ensures that the module is automatically loaded during system boot.

3 Running the ATLAS Modified SystecCAN Driver

This section describes how to run and utilize the ATLAS Modified SystecCAN Driver for the USB-CANmodul Series.

3.1 Starting the Driver

To start the driver, ensure the driver module is loaded into the kernel (if you’d like to have it load automatically, please refer to box 2-2). If the module is not loaded, use the following command:

```
sudo insmod systec_can.ko
```

This command does the following: - ‘sudo‘: Executes the command with superuser (root) privileges. - ‘insmod‘: Command to add a module to the Linux kernel. - ‘systec_can‘: The name of the driver module you want to load.

3.2 Configuring CAN Interfaces

To configure the CAN interface, use the `ip` command:

```
sudo ip link set <interface_name> up type can bitrate <bitrate>
```

Replace `<interface_name>` with the name of your CAN interface (e.g., ‘can0‘), and `<bitrate>` with your desired bitrate (e.g., ‘500000‘ for 500 kbps). This command sets up the CAN interface.

3.2.1 Example

Assuming your CAN interface is named ‘can0‘ and you want to set it up with a bitrate of 500,000 bits per second, you would run:

```
sudo ip link set can0 up type can bitrate 500000
```

Finding Your CAN Interface Name **3-1**

To find the name of your CAN interface, you can use the following command:

```
    ip link show
```

This command lists all network interfaces on your system. Look for the interface that corresponds to your CAN hardware, usually named ‘can0‘.

3.2.2

125000

3.3 Testing the CAN Communication

To verify CAN communication, use the `candump` utility to display incoming CAN messages on the specified interface:

```
candump <interface_name>
```

Replace `<interface_name>` with your actual CAN interface name (e.g., ‘can0‘). This command captures and displays all CAN frames received by the ‘can0‘ interface.

3.3.1 Example

If you have a CAN interface named ‘can0‘, running the following command will display all incoming CAN messages on that interface:

```
candump can0
```

To send CAN frames, use the `cansend` utility. For example:

```
cansend <interface_name> <can_id>#<data>
```

Replace `<interface_name>` with your actual CAN interface name (e.g., ‘can0‘), `<can_id>` with the CAN ID, and `<data>` with the data payload in hexadecimal format.

3.3.2 Example

To send a CAN frame with ID ‘123‘ and data ‘1122334455667788‘ on the ‘can0‘ interface, use:

```
cansend can0 123#1122334455667788
```

4 Usage Examples and Verification

This section provides practical examples and commands to verify the functionality of the ATLAS Modified SystecCAN Driver and the CAN interface.

4.1 Checking CAN Interface Status

To check the status of the CAN interface, use the following command:

```
ip link show <interface_name>
```

Replace `<interface_name>` with the name of your CAN interface (e.g., `can0`).

4.1.1 Example

To check the status of the `can0` interface, use:

```
ip link show can0
```

This command lists all network interfaces and their statuses. Look for an interface named `can0` (or the name of your CAN interface) in the output.

4.2 Sending and Receiving CAN Messages

4.2.1 Sending a CAN Message

To send a CAN message using the `cansend` utility, use the following command:

```
cansend <interface_name> <can_id>#<data>
```

Replace `<interface_name>` with your CAN interface name (e.g., `can0`), `<can_id>` with the CAN ID, and `<data>` with the data payload in hexadecimal format.

4.2.2 Example

To send a CAN message with ID `123` and data `DEADBEEF` on the `can0` interface, use:

```
cansend can0 123#DEADBEEF
```

4.2.3 Receiving CAN Messages

To receive CAN messages, use the `candump` utility:

```
candump <interface_name>
```

Replace `<interface_name>` with your CAN interface name (e.g., `can0`). This command will display all incoming CAN messages on the specified interface.

4.2.4 Example

To display all incoming CAN messages on the `can0` interface, use:

```
candump can0
```

4.3 Logging CAN Traffic

For logging CAN traffic to a file for later analysis, use:

```
candump <interface_name> > can_log.txt
```

Replace `<interface_name>` with your CAN interface name (e.g., `can0`). This command will save all incoming CAN messages to a file named `can_log.txt`.

4.3.1 Example

To log all incoming CAN messages on the `can0` interface to `can_log.txt`, use:

```
candump can0 > can_log.txt
```

4.4 Automated Testing Script

For automated testing, you can create a simple script. Below is an example of a Bash script to send and receive CAN messages:

```
#!/bin/bash
# Start candump in the background
candump can0 > can_log.txt &
CANDUMP_PID=$!

# Send a CAN message
cansend can0 123#DEADBEEF

# Wait for a few seconds
sleep 5

# Stop candump
kill $CANDUMP_PID

# Display log
cat can_log.txt
```

Save this script as `test_can.sh` and run it using:

```
chmod +x test_can.sh
./test_can.sh
```

4.4.1 Explanation

- ‘`#!/bin/bash`‘: Specifies the script should be run with the Bash shell.
- ‘`candump can0 > can_log.txt &`‘: Starts `candump` in the background, logging all messages to `can_log.txt`.
- ‘`CANDUMP_PID=$¡`: Saves the process ID of `candump` to a variable.
- ‘`cansend can0 123#DEADBEEF`‘: Sends a CAN message with ID `123` and data `DEADBEEF`.
- ‘`sleep 5`‘: Waits for 5 seconds.
- ‘`kill $CANDUMP_PID`‘: Stops `candump`.
- ‘`cat can_log.txt`‘: Displays the contents of `can_log.txt`.