User's Guide

AM62L-EVSE-DEV-EVM User Guide

TEXAS INSTRUMENTS

ABSTRACT

This document covers hardware connections, software configuration, and testing procedures for key interfaces including Power Line Communication (PLC), CAN bus, and UART (RS-485/RS-232) communications. It also explains how to configure and use the EVerest open-source charging stack, which provides a modular platform for EV charging infrastructure. Step-by-step instructions are provided for simulating an AC charging session according to IEC 61851-1 standards using the MSPM0 microcontroller, which handles critical analog handshakes and safety functions. The guide is designed to help developers accelerate EVSE development with reduced time-to-market across different charging standards and regional requirements.

Table of Contents

Trademarks

All trademarks are the property of their respective owners.

1 Introduction

Electric Vehicle Supply Equipment (EVSE) systems have become increasingly complex, requiring sophisticated control mechanisms to support multiple charging standards and protocols. The AM62L-EVSE-DEV-EVM reference design addresses this complexity by providing a comprehensive front-end controller solution that acts as the central communication module for the EV charging process [1].

This reference design combines the processing power of the AM62L processor with the MSPM0G3507 microcontroller to create a versatile platform supporting both AC and DC charging across global standards including Combined Charging System (CCS), Guobiao/Tuijian (GB/T), and Charge de Move (CHAdeMO) [1].

At the core of this system is the AM62L processor, which runs the EVerest Open Source Software charging stack on Linux, handling digital communication with electric vehicles. The AM62L supports Ethernet and wireless connectivity for backend communications, as well as display capabilities for human-machine interface (HMI) integration [1]. This processor enables developers to build sophisticated charging applications with a focus on user experience and system management.

Working alongside the AM62L, the MSPM0G3507 microcontroller serves as the front-end controller, managing the critical analog handshakes with electric vehicles and safety functions. The MSPM0 handles essential tasks like control pilot signaling, proximity detection, and monitoring of temperature sensors in the charging connectors [1]. This dual-processor approach ensures reliable operation of both high-level communication protocols and low-level safety mechanisms.

The design includes multiple communication interfaces—CAN, RS-485, RS-232, and Ethernet—enabling control of power conversion units, external metering devices, and other peripherals [1]. This comprehensive connectivity allows the system to integrate with various components in the charging infrastructure ecosystem.

By providing a complete reference platform with open-source software support, the AM62L-EVSE-DEV-EVM enables developers to accelerate the development of EVSE systems with reduced time-to-market and enhanced flexibility across different charging standards and regional requirements.

Please note that the AM62L-EVSE-DEV-EVM is also referred to as the TIDA-010939 in portions of this document. This is the reference design the AM62L-EVSE-DEV-EVM is based on.

2 Overview of the Software Drivers for an EVSE Development Platform

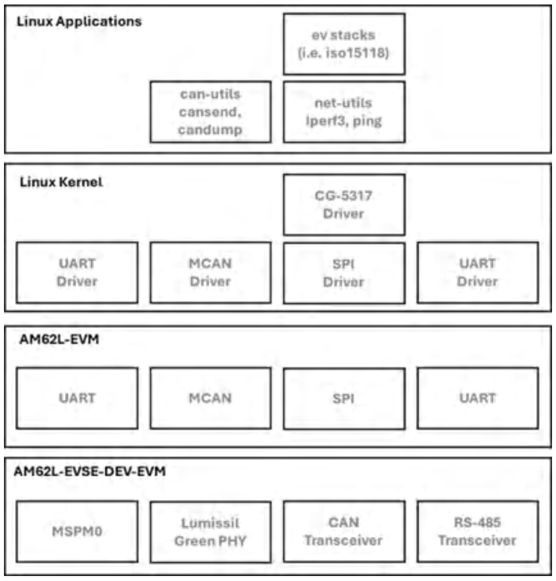


Figure 2-1. Linux Application to Hardware Flow

The diagram illustrates a layered architecture where:

Application Layer:

- Linux applications include networking utilities (net-utils like Iperf3, ping)
  - These utilities are provided in the image for the SD card as well as the TI Linux SDK.
- CAN utilities (can-utils including cansend, candump)
  - These utilities are provided in the image for the SD card as well as the TI Linux SDK.
- EV charging protocol stacks (ISO15118)

Driver Layer:

- The Linux Kernel contains several hardware drivers, the drivers used for this application include:
  - UART Driver
  - MCAN (Controller Area Network) Driver
  - SPI Driver
  - CG-5317 Driver

Hardware Layer:

- The system connects two main hardware platforms:
  - AM62L-EVM with UART, MCAN, UART, and SPI interfaces
  - AM62L-EVSE-DEV-EVM (Electric Vehicle Supply Equipment development board)

Physical Interface Layer:

- Hardware transceivers and PHY components including RS-485 Transceiver, CAN Transceiver, MSPM0, Lumissil, and Green PHY provide the physical interface to external systems [1]

The architecture demonstrates a typical Linux embedded system where applications make system calls to the kernel, which then uses appropriate device drivers to communicate with the underlying hardware interfaces. This enables high-level applications to control and communicate through various protocols (CAN, UART, SPI) without needing to directly manage the hardware complexity.

3 Recommended Test Setup for the AM62L-EVM and the AM62L-EVSE-DEV-EVM

To test the interfaces on the AM62L-EVSE-DEV-EVM the recommended setup is to have two AM62L EVMs connected with a AM62L-EVSE-DEV-EVM board. This setup would enable users to be able to test without requiring expensive test equipment. For example, Power Line Communications (PLC) of the Lumissil Green PHY device that is on the AM62L-EVSE-DEV-EVM can be tested using ethernet like IP networking. The CANBUS can be tested with either the other AM62L/AM62L-EVSE-DEV-EVM using the on board canutils application or an external CANBUS analyzer. Finally, the RS-485 and RS-232 connection can be tested with the AM62L/AM62L-EVSE-DEV-EVM using stty between the two boards.

The suggested test setup is shown in the following diagram.

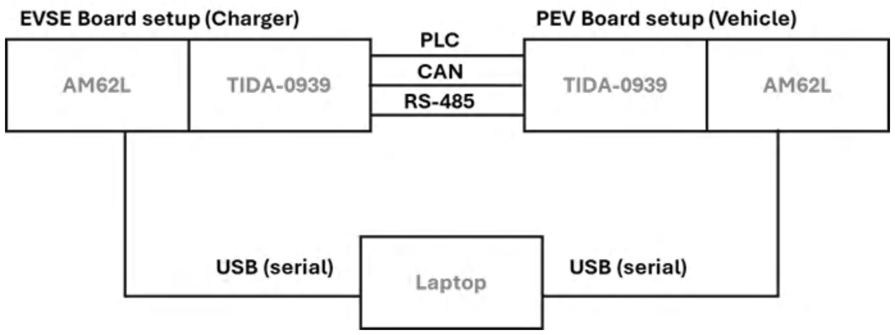


Figure 3-1. Suggested Test Setup

Basic operational testing of the PLC interface can be tested using network tools provided in the image for the board. These utilities are standard in the TI Linux SDK as well. When testing the PLC interface, one board will need to be designated as the EVSE and the other PEV. As part of this designation there are different configurations for the PLC endpoints that enable the SLAC process to complete. Later in the document there will be a step-by-step guide for the configuration steps and the operation of the interface between two boards. The CAN and RS-485 / RS-232 interfaces will also have a step-by-step guide later in the document.

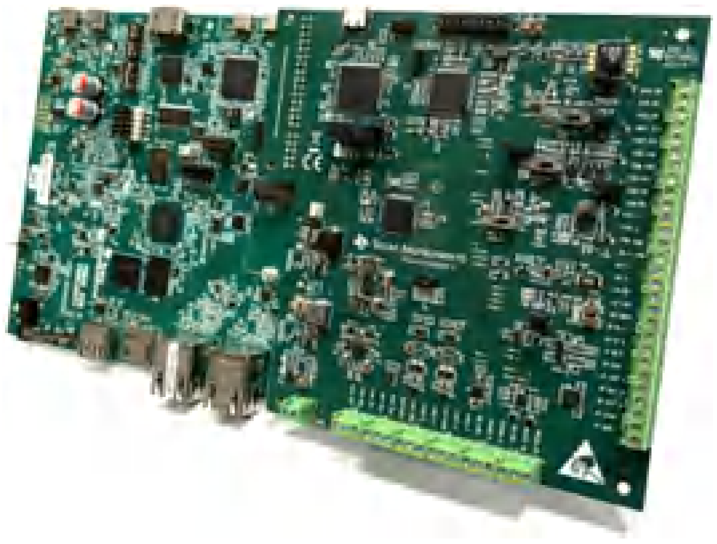
The laptop in the diagram can be of your choice (Linux, Windows, MAC). The requirement will be that you are connected to USB serial connections on the AM62L EVMs.

The test setup will allow you to become familiar with the AM62L with AM62L-EVSE-DEV-EVM board combination without having to have or use expensive test equipment for PLC, CAN and RS-485/RS-232.

4 Software Out of Box with the AM62L-EVM and the AM62L-EVSE-DEV-EVM

This section will help you get started with your AM62L evaluation module (EVM) using the Linux SDK and the AM62L-EVSE-DEV-EVM EV Charging Reference design for developing EV charging or also known as EV Supply Equipment (EVSE). To begin, please follow the out of box or quick start process for the AM62L EVM. Once that step is completed you will then add in the device tree overlay support for the AM62L-EVSE-DEV-EVM board.

It is assumed that the AM62L EVM and the AM62L-EVSE-DEV-EVM boards have been connected to one another.



As mentioned earlier in this document it is recommended that to get the best experience there should be a two-board setup.

Please note that reading the AM62L-EVM Quick start guide will ask you to download an image. This link will be the wic image to program to the SD card:

Please note that the image has the EVerest application in the image and is defaulted to run at start-up. Several of the test sections in the guide will provide instructions on how to stop the EVerest application so that interface testing demonstrations can be performed.

Please review the AM62L-EVM quick start guide located here:

https://dev.ti.com/tirex/local?id=TMDS62LEVM-QSG&packageId=PROCESSORS-DEVTOOLS

Please note that the quick start guide describes several steps, these are the only steps that will be used:

- Power supply necessary, USB type-C and the current requirements.
- How to attach a serial console and the type of cable needed
- Boot mode selection (this application will be using SD card and is the default boot mode of the AM62L EVM)
- How to program a wic image onto the SD card
- The serial console that is necessary and how to enable one on your computer
- Ethernet (optional, this interface is not necessary in this application).
- How to log into the EVM after it has been booted.

The following steps will not be used in this application:

- HDMI
- Mouse
- Running the other demos mentioned in the quick start guide.

The quick start guide also provides some basic debugging steps if the EVM does not boot.

The wic image provided will have the necessary driver support for the AM62L-EVSE-DEV-EVM board as the image already has the necessary support for the AM62L-EVSE-DEV-EVM daughterboard. This section is to show how to add the necessary DT support for the AM62L-EVSE-DEV-EVM board if you are building your own distribution.

For the Linux kernel to be able to access the interface on the AM62L-EVSE-DEV-EVM support for the AM62L-EVSE-DEV-EVM is added to a file called uEnv.txt located in the boot partition of the SD card. Use a USB SD Card reader and insert the reader into the Linux machine or Windows machine. You will edit the uEnv.txt file located in the boot partition.

This file k3-am62l3-evm-tida-010939.dtbo will need to be added to the name_overlays line in the file.

This section in the SDK user's guide provides more background on adding overlays to the uEnv.txt file.

https://software-dl.ti.com/processor-sdk-linux/esd/AM62LX/11_01_16_13/exports/docs/linux/How_to_Guides/Target/How_to_enable_DT_overlays_in_linux.html

Additional Resources:

- TIDA-010939 Design Guide
- Link to the TIDA-010939 HW user guide
- AM62L Technical Reference Manual
- Linux SDK User Guide
- MSPM0-SDK
- TI E2E Support Forum

5 EVerest: Open Source Electric Vehicle Charging Infrastructure

Overview

EVerest is an open source software stack for EV (Electric Vehicle) charging infrastructure developed by Pionix GmbH. It represents a modular, flexible platform designed to address the growing needs of the electric mobility industry.

Key Features

- Modular Architecture: Built with a component-based design that allows developers to easily customize and extend functionality
- Open Source: Released under Apache 2.0 license, fostering collaboration and innovation
- Hardware Agnostic: Works across different charging hardware platforms
- Interoperable: Supports industry standards like OCPP (Open Charge Point Protocol)
- Scalable: Suitable for both simple residential chargers and complex commercial charging stations

Applications

EVerest serves as the foundation for various charging solutions:

- Home charging stations
- Public charging infrastructure
- Fleet charging management
- Smart charging with renewable energy integration
- Vehicle-to-grid (V2G) applications

Benefits

- Reduced Development Time: Pre-built modules accelerate time-to-market
- Future-Proof: Modular design allows for easy updates as technology evolves
- Vendor Independence: Avoids proprietary lock-in
- Cost Efficiency: Leverages open source to reduce development costs
- Community Support: Backed by a growing ecosystem of developers and industry partners

About Pionix

Pionix GmbH is the company behind EVerest, focused on accelerating the transition to sustainable mobility through open source solutions. They provide commercial support, consulting, and custom development services related to the EVerest platform.

6 Testing PLC Communications

The best test method for testing the power-line-communications (PLC) of the AM62L and AM62L-EVSE-DEV-EVM board combination is to connect two AM62L EVMs with respective AM62L-EVSE-DEV-EVM boards together. This method for a test setup will be the easiest way to demonstrate communication with a PLC link partner. This section is written to comprehend this type of setup. This is the diagram from the recommended setup section provided as a quick reference showing the two-board setup.

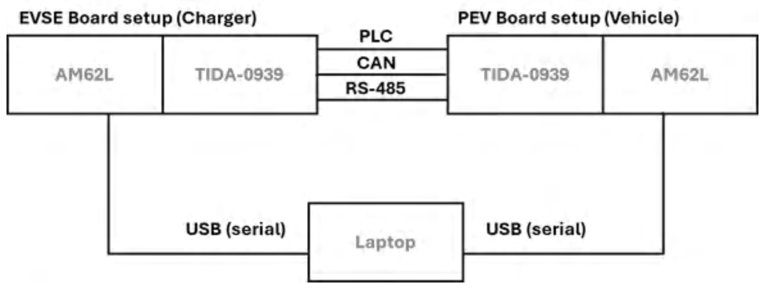


Figure 6-1. Two-board Setup

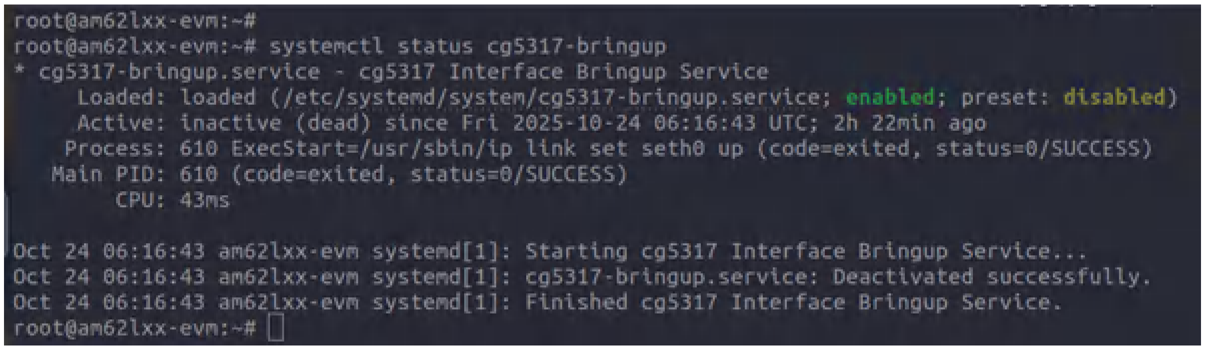
The board setup on the left is considered the Charger and the board setup on the right is considered the EV. As you read this section you will be provided with the context that the step that is being executed on, EVSE or PEV.

The following instructions require the SD card image defined in the out of box section of this document. As pointed out earlier in the document, there is a device tree source overlay (dtso) file that describes the interfaces on the AM62L-EVSE-DEV-EVM board that has been provided as a device tree binary (dtbo). This dtbo for the AM62L-EVSE-DEV-EVM board has the necessary PLC interface configurations. This interface will be on the spi0 interface that will become the seth0 interface for communication. The code assumes that the dtbo has been added to the uEnv.txt file in the boot partition of the SD card. This file is read by U-Boot during the system boot sequence.

To enable the Lumissil PLC it must first be powered and after that reset. The AM62L software is configured to reset the PLC before SPI communication is enabled. Therefore, it is important to power up the AM62L-EVSE-DEV-EVM (which hosts the PLC) before starting the AM62L-EVM.

The first step to verifying CG-5317 functionality is to verify that the driver has initialized. Use these commands to verify CG-5317 has been configured correctly. The command will check the bring up status of a systemd service that is for the Lumissil driver. The screen capture shows a successful enabling of the interface. This step should be performed on both the EVSE and PEV board setups.

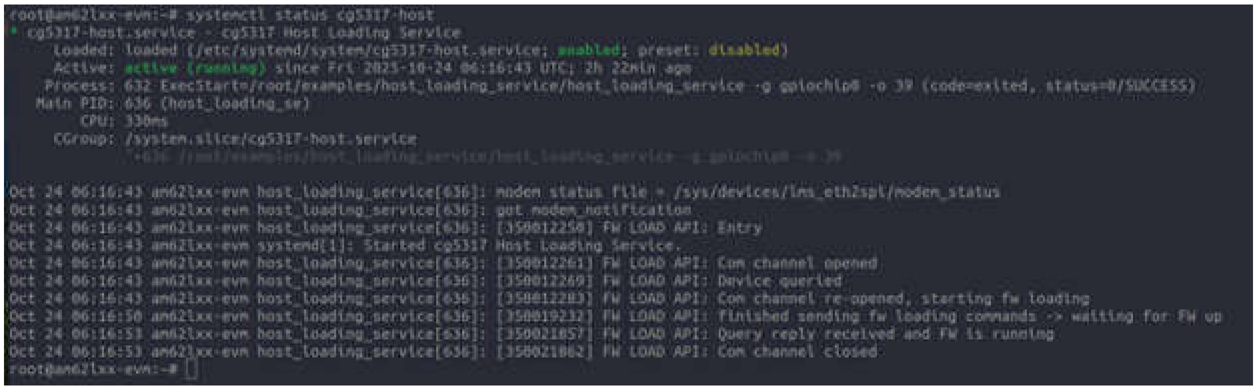
```
root@am62lxx-evm:~# systemctl status cg5317-bringup
```



```
root@am62lxx-evm:~#
root@am62lxx-evm:~# systemctl status cg5317-bringup
* cg5317-bringup.service - cg5317 Interface Bringup Service
     Loaded: loaded (/etc/systemd/system/cg5317-bringup.service; enabled; preset: disabled)
     Active: inactive (dead) since Fri 2025-10-24 06:16:43 UTC; 2h 22min ago
    Process: 610 ExecStart=/usr/sbin/ip link set seth0 up (code=exited, status=0/SUCCESS)
   Main PID: 610 (code=exited, status=0/SUCCESS)
        CPU: 43ms

Oct 24 06:16:43 am62lxx-evm systemd[1]: Starting cg5317 Interface Bringup Service...
Oct 24 06:16:43 am62lxx-evm systemd[1]: cg5317-bringup.service: Deactivated successfully.
Oct 24 06:16:43 am62lxx-evm systemd[1]: Finished cg5317 Interface Bringup Service.
root@am62lxx-evm:~#
```

After the bring-up check, the next step is to verify that the interface has been initialized.

```
root@am62lxx-evm:~# systemctl status cg5317-host
```

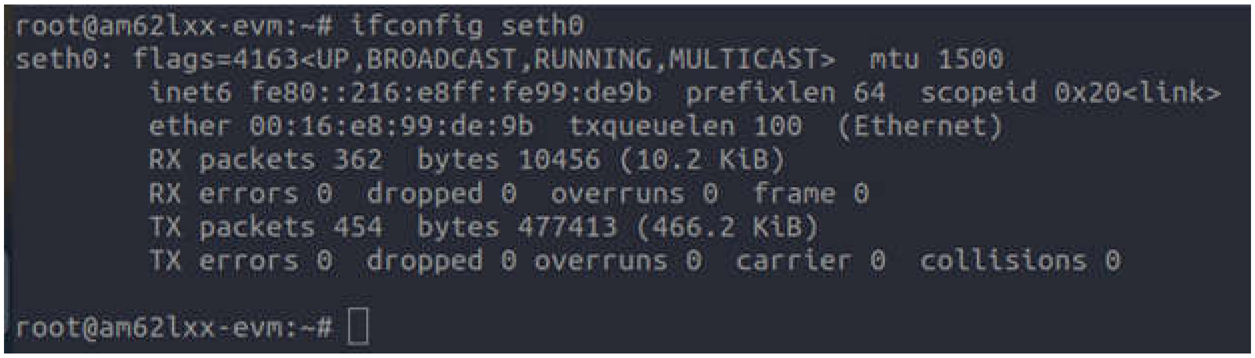


```
root@am62lxx-evm:~# systemctl status cg5317-host
* cg5317-host.service - cg5317 Host Loading Service
     Loaded: loaded (/etc/systemd/system/cg5317-host.service; enabled; preset: disabled)
     Active: active (running) since Fri 2025-10-24 06:16:43 UTC; 2h 22min ago
    Process: 632 ExecStart=/root/examples/host_loading_service/host_loading_service -g gpiochip0 -o 39 (code=exited, status=0/SUCCESS)
   Main PID: 636 (host_loading_se)
        CPU: 330ms
     CGroup: /system.slice/cg5317-host.service
             └─636 /root/examples/host_loading_service/host_loading_service -g gpiochip0 -o 39

Oct 24 06:16:43 am62lxx-evm host_loading_service[636]: modem status file = /sys/devices/lms_eth2spi/modem_status
Oct 24 06:16:43 am62lxx-evm host_loading_service[636]: got modem_notification
Oct 24 06:16:43 am62lxx-evm host_loading_service[636]: [350012258] FW LOAD API: Entry
Oct 24 06:16:43 am62lxx-evm systemd[1]: Started cg5317 Host Loading Service.
Oct 24 06:16:43 am62lxx-evm host_loading_service[636]: [350012261] FW LOAD API: Com channel opened
Oct 24 06:16:43 am62lxx-evm host_loading_service[636]: [350012269] FW LOAD API: Device queried
Oct 24 06:16:43 am62lxx-evm host_loading_service[636]: [350012283] FW LOAD API: Com channel re-opened, starting fw loading
Oct 24 06:16:50 am62lxx-evm host_loading_service[636]: [350019232] FW LOAD API: finished sending fw loading commands -> waiting for FW up
Oct 24 06:16:53 am62lxx-evm host_loading_service[636]: [350021857] FW LOAD API: Query reply received and FW is running
Oct 24 06:16:53 am62lxx-evm host_loading_service[636]: [350021862] FW LOAD API: Com channel closed
root@am62lxx-evm:~# 
```

The above diagram shows that the cg5317 interface has been initialized successfully.

The last step of confirming a successful cg5317 initialization is verify that the interface is up and in the network interface list with this command.

root@am62lxx-evm:~# ifconfig seth0



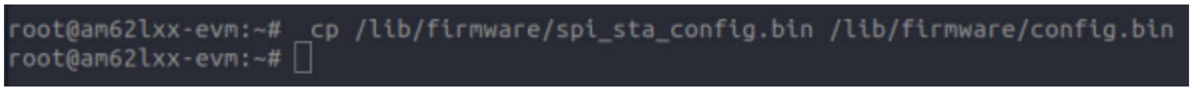
```
root@am62lxx-evm:~# ifconfig seth0
seth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::216:e8ff:fe99:de9b  prefixlen 64  scopeid 0x20<link>
        ether 00:16:e8:99:de:9b  txqueuelen 100  (Ethernet)
        RX packets 362  bytes 10456 (10.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 454  bytes 477413 (466.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@am62lxx-evm:~# 
```

If these steps are successful, then you are ready to move on to configuration for testing communication.

For a link to be set up between an EVSE and EV and a script must be run that configures one board to be the EVSE and the board combination as an EV. Like the block diagram above choose one board combination to be the EVSE and the other board to be the EV. We will call it PEV from now on to match the binaries in the file system.

To configure the PEV board perform these commands:

root@am62lxx-evm:~# cp /lib/firmware/spi_sta_config.bin /lib/firmware/config.bin

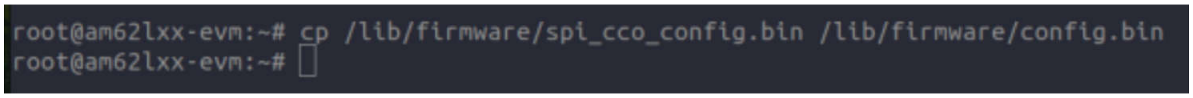


```
root@am62lxx-evm:~#  cp /lib/firmware/spi_sta_config.bin /lib/firmware/config.bin
root@am62lxx-evm:~# 
```

root@am62lxx-evm:~# reboot

To configure the EVSE board perform this command:

root@am62lxx-evm:~# cp /lib/firmware/spi_cco_config.bin /lib/firmware/config.bin



```
root@am62lxx-evm:~# cp /lib/firmware/spi_cco_config.bin /lib/firmware/config.bin
root@am62lxx-evm:~# 
```

root@am62lxx-evm:~# reboot

After the reboot is complete

On the PEV board run:

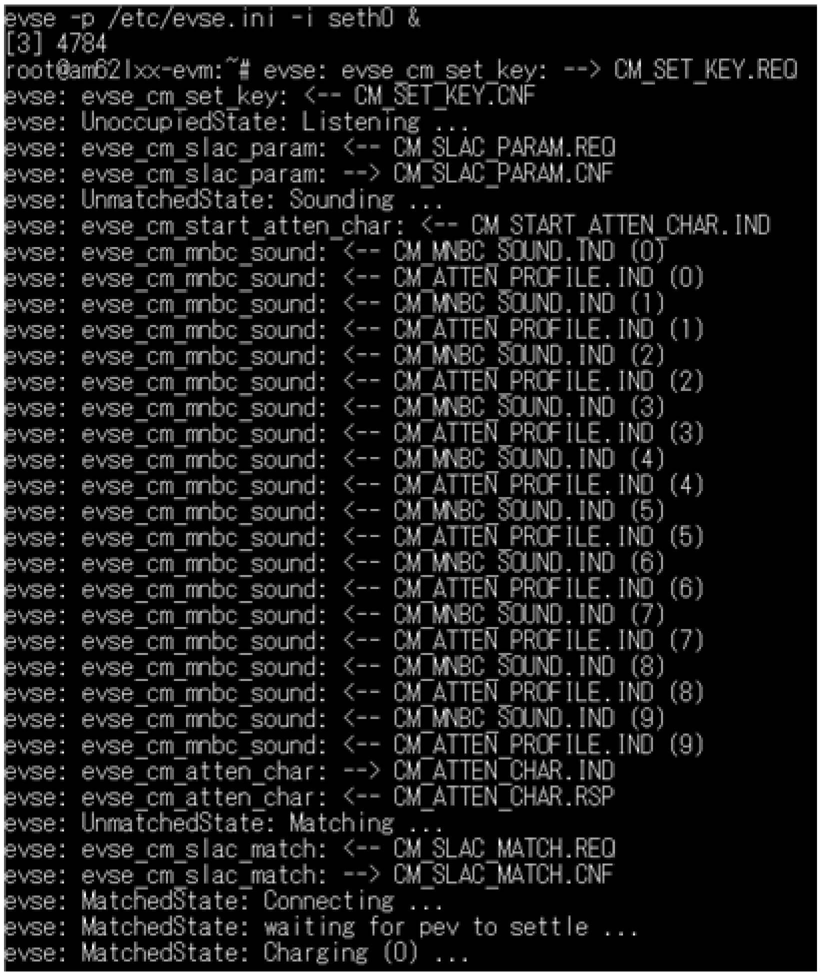
root@am62lxx-evm:~# pev -p /etc/pev.ini -i seth0 &

On the EVSE board run:

root@am62lxx-evm:~# evse -p /etc/evse.ini -i seth0 &

These commands will initiate the SLAC sequence between the PEV and EVSE boards. SLAC (Signal Level Attenuation Characterization) is a protocol used in power line communications (PLC), particularly in standards like HomePlug Green PHY. This initialization process is required to establish a connection between the EVSE and PEV devices communicating over the cable connecting the two systems.

Once this process is completed the interface will be ready for IP networking. You should see messages saying charging on both the EVSE and PEV platforms after the SLAC process is completed. This is shown in the picture below.



```
evse -p /etc/evse.ini -i seth0 &
[3] 4784
root@am62lxx-evm:~# evse: evse_cm_set_key: --> CM_SET_KEY.REQ
evse: evse_cm_set_key: <-- CM_SET_KEY.CNF
evse: UnoccupiedState: Listening ...
evse: evse_cm_slac_param: <-- CM_SLAC_PARAM.REQ
evse: evse_cm_slac_param: --> CM_SLAC_PARAM.CNF
evse: UnmatchedState: Sounding ...
evse: evse_cm_start_atten_char: <-- CM_START_ATTEN_CHAR.IND
evse: evse_cm_mnbc_sound: <-- CM_MNBC_SOUND.IND (0)
evse: evse_cm_mnbc_sound: <-- CM_ATTEN_PROFILE.IND (0)
evse: evse_cm_mnbc_sound: <-- CM_MNBC_SOUND.IND (1)
evse: evse_cm_mnbc_sound: <-- CM_ATTEN_PROFILE.IND (1)
evse: evse_cm_mnbc_sound: <-- CM_MNBC_SOUND.IND (2)
evse: evse_cm_mnbc_sound: <-- CM_ATTEN_PROFILE.IND (2)
evse: evse_cm_mnbc_sound: <-- CM_MNBC_SOUND.IND (3)
evse: evse_cm_mnbc_sound: <-- CM_ATTEN_PROFILE.IND (3)
evse: evse_cm_mnbc_sound: <-- CM_MNBC_SOUND.IND (4)
evse: evse_cm_mnbc_sound: <-- CM_ATTEN_PROFILE.IND (4)
evse: evse_cm_mnbc_sound: <-- CM_MNBC_SOUND.IND (5)
evse: evse_cm_mnbc_sound: <-- CM_ATTEN_PROFILE.IND (5)
evse: evse_cm_mnbc_sound: <-- CM_MNBC_SOUND.IND (6)
evse: evse_cm_mnbc_sound: <-- CM_ATTEN_PROFILE.IND (6)
evse: evse_cm_mnbc_sound: <-- CM_MNBC_SOUND.IND (7)
evse: evse_cm_mnbc_sound: <-- CM_ATTEN_PROFILE.IND (7)
evse: evse_cm_mnbc_sound: <-- CM_MNBC_SOUND.IND (8)
evse: evse_cm_mnbc_sound: <-- CM_ATTEN_PROFILE.IND (8)
evse: evse_cm_mnbc_sound: <-- CM_MNBC_SOUND.IND (9)
evse: evse_cm_mnbc_sound: <-- CM_ATTEN_PROFILE.IND (9)
evse: evse_cm_atten_char: --> CM_ATTEN_CHAR.IND
evse: evse_cm_atten_char: <-- CM_ATTEN_CHAR.RSP
evse: UnmatchedState: Matching ...
evse: evse_cm_slac_match: <-- CM_SLAC_MATCH.REQ
evse: evse_cm_slac_match: --> CM_SLAC_MATCH.CNF
evse: MatchedState: Connecting ...
evse: MatchedState: waiting for pev to settle ...
evse: MatchedState: Charging (0) ...
```

Now IP addresses need to be assigned on both the EVSE and the PE boards. For the PEV board run this command:

ifconfig seth0 192.168.1.2/24 up

For the EVSE board run this command:

ifconfig seth0 192.168.1.1/24 up

Now the boards are ready to do ip networking:

From the PEV board run:

ping 192.168.1.1 (after a few successful pings ctrl-c to exit)

From the EVSE board run:

ping 192.168.1.2 (after a few successful pings ctrl-c to exit)

Now run the iperf command to test moving data across the link

On the EVSE run the iperf3 server command:

iperf3 -s -i 2

On the PEV board run the iperf3 client command:

iperf3 -c 192.168.1.1 -t 20

After 20 seconds the iperf3 client command on the PEV will stop. You should see a network throughput that averages less than 1Mbps.

This completes the test.

7 Testing CAN Communications

It is suggested that two AM62L with the companion AM62L-EVSE-DEV-EVM board wire the main_mcan0 interfaces together. Like the other sections in this document, the tests will be performed from an EVSE and PEV perspective as shown in this diagram.

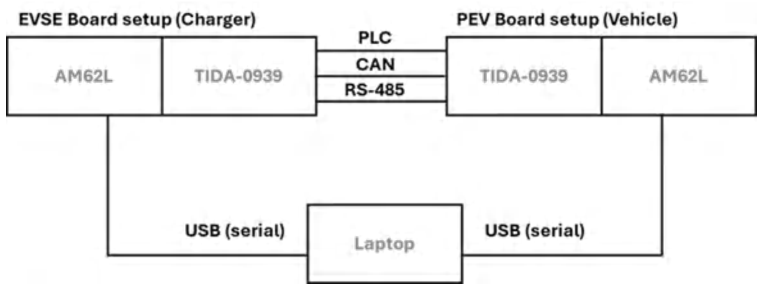


Figure 7-1. EVSE and PEV Perspectives

Some background first is to note that the default SDK comes with the MCAN support configured into the Linux kernel. The next thing to point out is that MCAN interfaces need to be enabled in the device tree source file. As pointed out earlier in the document, there is a device tree source overlay (dtso) file that describes the interfaces on the AM62L-EVSE-DEV-EVM board that has been provided as device tree binary (dtbo). This dtbo for the AM62L-EVSE-DEV-EVM board has the necessary MCAN interface configurations. The code assumes that the dtbo has been added to the uEnv.txt file in the boot partition of the SD card. This file is read by U-Boot during the system boot sequence.

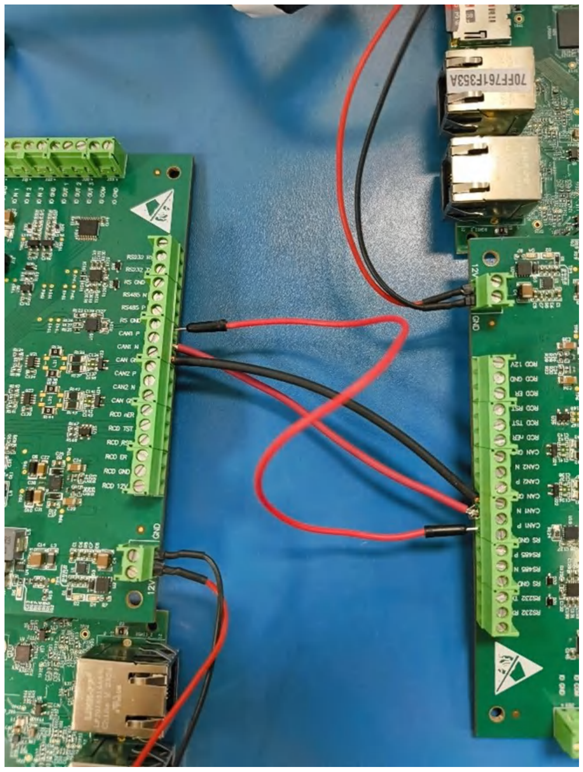


Figure 7-2. Connecting the Two Board CAN Interfaces

Please note the wire that is used to connect the two boards. Please be aware that when connecting CAN interfaces require termination resistors which are already on the lines between the connector and the CAN transceiver on the board.

The CAN1 transceiver on the AM62L-EVSE-DEV-EVM is directly connected to the main_mcan1 interface of the AM62L. To establish a connection between a second CAN interface of the AM62L and the second transceiver on the AM62L-EVSE-DEV-EVM, a wire must be connected between both boards.

The AM62L MCAN0 interface is routed to the J16 header located on the EVM. To connect the MCAN0 interface to the transceiver the following pins should be connected:

Table 7-1. J16 Pins

TIDA-010939 - J29		AM62L-EVM - J16	Function
Pin 1	To	Pin 2	TX
Pin 2	To	Pin 3	RX
Pin 3	To	Pin 4	GND

If the MCAN2 interface of the AM62L should be used, connect J18 of the EVM with the CAN2 header on the AM62L-EVSE-DEV-EVM:

Table 7-2. J18 Pins

TIDA-010939 - J29		AM62L-EVM - J18	Function
Pin 1	To	Pin 2	TX
Pin 2	To	Pin 3	RX
Pin 3	To	Pin 4	GND

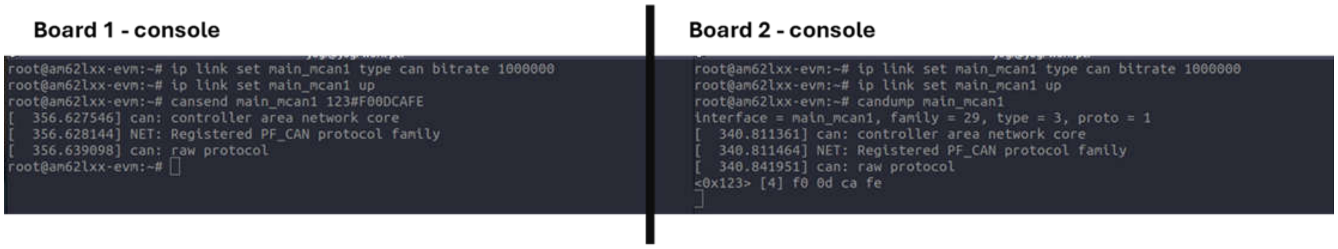
Please refer to this link for more additional information on the MCAN driver in the TI Linux SDK. https://software-dl.ti.com/processor-sdk-linux/esd/AM62X/11_01_05_03/exports/docs/linux/Foundational_Components/Kernel/Kernel_Drivers/MCAN.html

The MCAN Interface configuration happens using network utilities called ip commands which are for managing various aspects of a system's network configuration. Please note that the MCAN interface is part of the system's network stack. These commands are in the default file system of the TI Linux SDK.

The MCAN interface can be configured in one of two different modes, CAN and CAN-FD. For CAN mode perform this step:

ip link set main_mcan0 type can bitrate 1000000

The screen capture below shows the sequence of initializing the initialization for CAN and then checking the interface list for the can2 interface and its status. There are two methods shown here, ip-route2 based and ifconfig, net-utils based. Either method is fine to use for checking interface status.



Board 1 - console

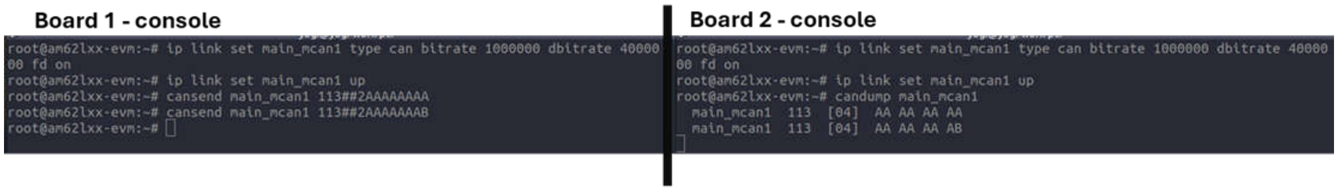
```
root@am62lxx-evm:~# ip link set main_mcan1 type can bitrate 1000000
root@am62lxx-evm:~# ip link set main_mcan1 up
root@am62lxx-evm:~# cansend main_mcan1 123#F00DCAFE
[  356.627546] can: controller area network core
[  356.628144] NET: Registered PF_CAN protocol family
[  356.639098] can: raw protocol
root@am62lxx-evm:~# 
```

Board 2 - console

```
root@am62lxx-evm:~# ip link set main_mcan1 type can bitrate 1000000
root@am62lxx-evm:~# ip link set main_mcan1 up
root@am62lxx-evm:~# candump main_mcan1
interface = main_mcan1, family = 29, type = 3, proto = 1
[  340.811361] can: controller area network core
[  340.811464] NET: Registered PF_CAN protocol family
[  340.841951] can: raw protocol
<0x123> [4] f0 0d ca fe
```

For CAN-FD perform this step:

ip link set main_mcan0 type can bitrate 1000000 dbitrate 4000000 fd on



Board 1 - console

```
root@am62lxx-evm:~# ip link set main_mcan1 type can bitrate 1000000 dbitrate 40000
00 fd on
root@am62lxx-evm:~# ip link set main_mcan1 up
root@am62lxx-evm:~# cansend main_mcan1 113##2AAAAAAAA
root@am62lxx-evm:~# cansend main_mcan1 113##2AAAAAAAB
root@am62lxx-evm:~# 
```

Board 2 - console

```
root@am62lxx-evm:~# ip link set main_mcan1 type can bitrate 1000000 dbitrate 40000
00 fd on
root@am62lxx-evm:~# ip link set main_mcan1 up
root@am62lxx-evm:~# candump main_mcan1
  main_mcan1  113  [04]  AA AA AA AA
  main_mcan1  113  [04]  AA AA AA AB
```

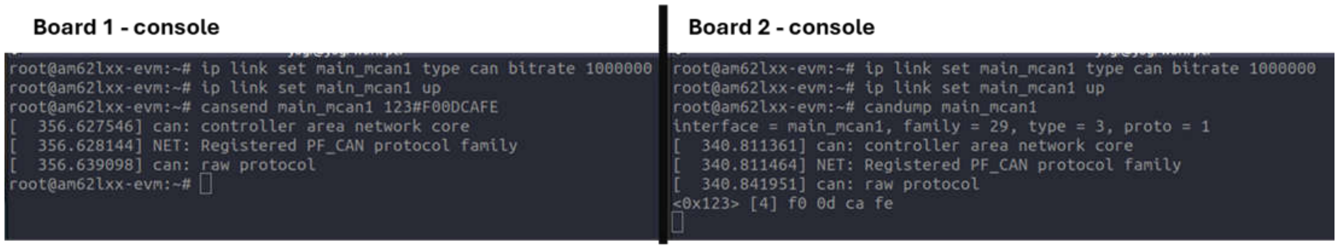
After initializing the interface an ip command is used to bring up the interface.

ip link set main_mcan0 up

To send frames the following commands would be used. For sending frames in CAN mode:

cansend main_mcan0 123#F00DCAFE

Notice the Board 2 console candump output matches what was sent using cansend from Board 1.



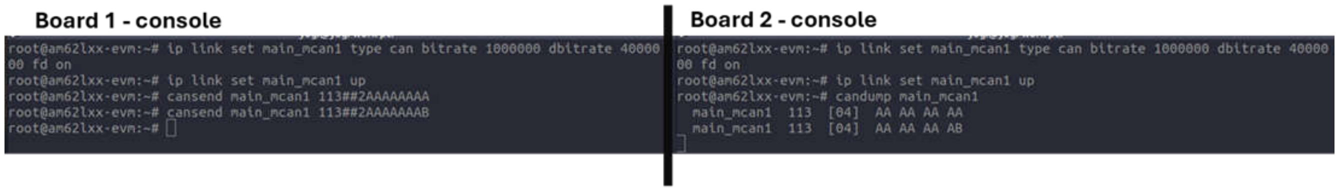
```
root@am62lxx-evm:~# ip link set main_mcan1 type can bitrate 1000000
root@am62lxx-evm:~# ip link set main_mcan1 up
root@am62lxx-evm:~# cansend main_mcan1 123#F00DCAFE
[  356.627546] can: controller area network core
[  356.628144] NET: Registered PF_CAN protocol family
[  356.639098] can: raw protocol
root@am62lxx-evm:~#
```

```
root@am62lxx-evm:~# ip link set main_mcan1 type can bitrate 1000000
root@am62lxx-evm:~# ip link set main_mcan1 up
root@am62lxx-evm:~# candump main_mcan1
interface = main_mcan1, family = 29, type = 3, proto = 1
[  340.811361] can: controller area network core
[  340.811464] NET: Registered PF_CAN protocol family
[  340.841951] can: raw protocol
<0x123> [4] f0 0d ca fe
```

For sending frames in CAN-FD mode

cansend main_mcan0 113##2AAAAAAAA

candump main_mcan0



```
root@am62lxx-evm:~# ip link set main_mcan1 type can bitrate 1000000 dbitrate 40000
00 fd on
root@am62lxx-evm:~# ip link set main_mcan1 up
root@am62lxx-evm:~# cansend main_mcan1 113##2AAAAAAAA
root@am62lxx-evm:~# cansend main_mcan1 113##2AAAAAAAB
root@am62lxx-evm:~#
```

```
root@am62lxx-evm:~# ip link set main_mcan1 type can bitrate 1000000 dbitrate 40000
00 fd on
root@am62lxx-evm:~# ip link set main_mcan1 up
root@am62lxx-evm:~# candump main_mcan1
  main_mcan1  113  [04]  AA AA AA AA
  main_mcan1  113  [04]  AA AA AA AB
```

Notice the Board 2 console candump output matches what was sent using cansend from Board 1.

Additional information can be found in the above link to the MCAN Linux Kernel user guide in the above link.

8 Testing UART Communications AM62L-EVM and the AM62L-EVSE-DEV-EVM

The best test method for testing the UART communications of the AM62L and AM62L-EVSE-DEV-EVM board combination is to connect two AM62L EVMs with respective AM62L-EVSE-DEV-EVM boards together. This method for a test setup will be easiest way to demonstrate communication with a UART link partner. This section is written to comprehend this type of setup. This is the diagram from the recommended setup section provided as a quick reference showing the two-board setup.

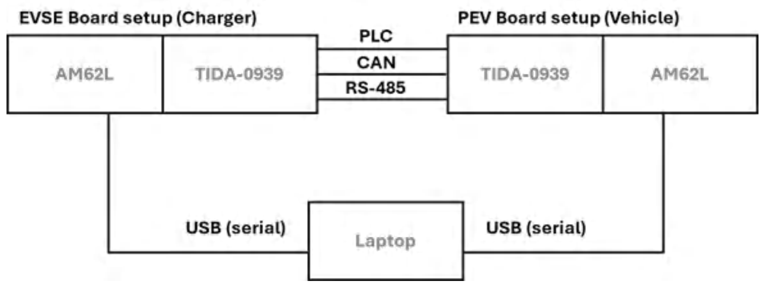


Figure 8-1. Two-board Setup

For RS485 Testing:

Take 2 AM62L + AM62L-EVSE-DEV-EVM boards (we will call them Board A & Board B). Wire the connections between the 2 boards as per the table below.

Table 8-1. RS485 Testing

Board A		Board B
RS485 N	ßà	RS485 N
RS485 P	ßà	RS485 P
RS GND	ßà	RS GND

See the following diagram for more details.

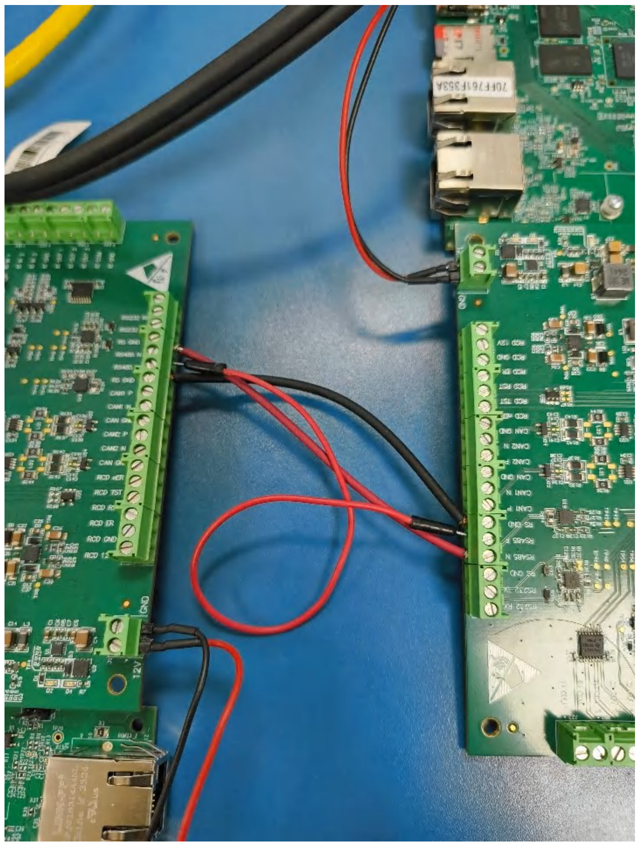


Figure 8-2. RS485 Testing

For RS232 Testing:

Take 2 AM62L + AM62L-EVSE-DEV-EVM boards (we will call them Board A & Board B). Wire the connections between the 2 boards as per the table below.

Table 8-2. RS232 Testing

Board A		Board B
RS232 TX	ßà	RS232 RX
RS232 RX	ßà	RS232 Tx
RS GND	ßà	RS GND

See the following diagram for more details.

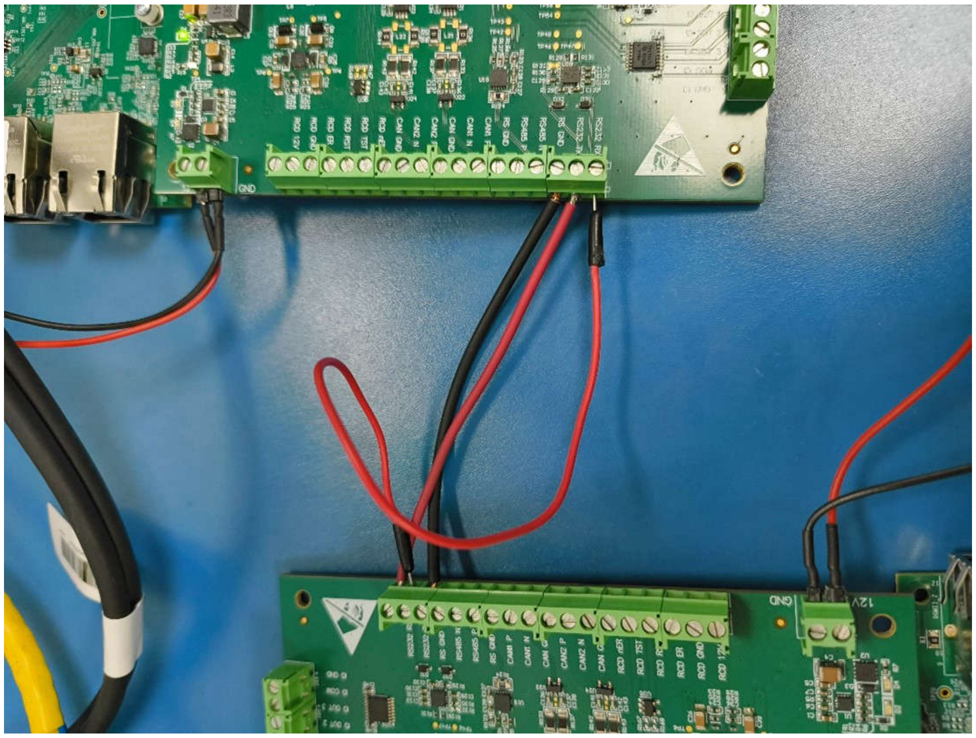


Figure 8-3. RS232 Testing

As pointed out earlier in the document, there is a device tree source overlay (dtso) file has that describes the interfaces on the AM62L-EVSE-DEV-EVM board has been provided as device tree binary (dtbo). This dtbo for the AM62L-EVSE-DEV-EVM board has the necessary UART interface configurations. The code assumes that the dtbo has been added to the uEnv.txt file in U-Boot.

Testing:

For RS485 the UART port on Linux is ttyS4

For RS232 the UART port on Linux is ttyS3

Replace the <name-of-the-port> in the commands given below to appropriate UART port as per your tests.

Use stty command to configure the UART interface on each board

stty -F /dev/<name-of-th-port> 9600 cs8 -cstopb -parenb raw -echo

Use the commands below to test

On Board A:

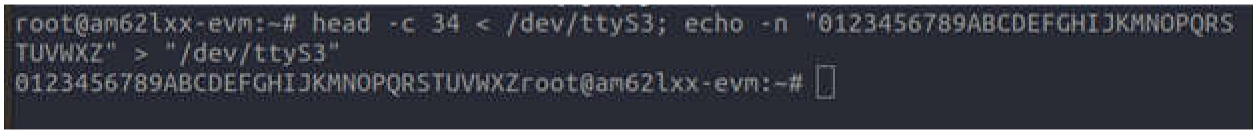
head -c 34 < /dev/<name-of-th-port>; echo -n "0123456789ABCDEFGHIJKMNOPQRSTUVWXZ" > /dev/<name-of-th-port>

On Board B:

echo -n "0123456789ABCDEFGHIJKMNOPQRSTUVWXZ" > /dev/<name-of-th-port>; head -c 34 < /dev/<name-of-th-port>

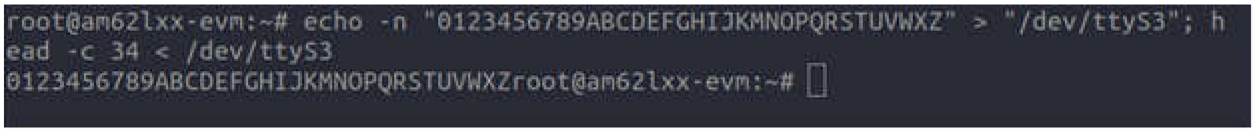
You should see the data being sent and read on both the boards. See the image below:

Board A:



```
root@am62lxx-evm:~# head -c 34 < /dev/ttyS3; echo -n "0123456789ABCDEFGHIJKMNOPQRS
TUVWXZ" > "/dev/ttyS3"
0123456789ABCDEFGHIJKMNOPQRSTUVWXZroot@am62lxx-evm:~#
```

Board B:



```
root@am62lxx-evm:~# echo -n "0123456789ABCDEFGHIJKMNOPQRSTUVWXZ" > "/dev/ttyS3"; h
ead -c 34 < /dev/ttyS3
0123456789ABCDEFGHIJKMNOPQRSTUVWXZroot@am62lxx-evm:~# 
```

If the data being sent and received is not the same then the test is failed.

9 Lab Testing Setup for the MSPM0

This test simulates an AC charging session according to the IEC 61851-1 standard by using the EV-simulation circuitry on the AM62L-EVSE-DEV-EVM and explains basic configurations of EVerest.

Hardware Setup

- Connect the AM62L-EVM to the AM62L-EVSE-DEV-EVM.
- Connect a 12V power supply to the screw terminal J1 on the AM62L-EVSE-DEV-EVM.
- Connect USB-C cable between the AM62L-EVSE-DEV-EVM and test PC.
- Connect a USB-Micro cable between the AM62L-EVM J7 and the test PC for serial communication.
- Connect a USB-C power supply to the AM62L Header J17.
- Connect a jumper on J5 between pin 1 & pin 2 (EVSE) to connect the PWM output to the screw terminal J4.
- Connect a jumper wire between the screw terminal J4 Pin 3 (CCS CP) and pin header J5 pin 3. This wire connects the Control Pilot signal to the EV-Simulation circuit.
- Set switch S1 towards the right, in direction of pin 3, to momentarily disconnect the Control Pilot signal from the EV simulation circuitry.
- Connect a jumper wire to the screw terminal J21 Pin 2 (IO IN 1). This jumper will later be used to enable charging state C – EV requests power by connecting it to a logic high, e.g. J2 Pin 2 (5V).
- Connect a resistor between screw terminal J4 pin 1 (PP) and screw terminal J4 pin 2 (PE) to indicate a charging cable is connected. The value of the resistor will indicate the maximum charging current capability of the charging cable. A 1.5kΩ resistor will allow currents up to 13A. Jumper J6 must be set.
- Optional an oscilloscope probe can be connected to the control pilot signal (TP19).
- The PLC needs to be reset after power up. Therefore, the AM62L-EVSE-DEV-EVM must be powered before the AM62L.

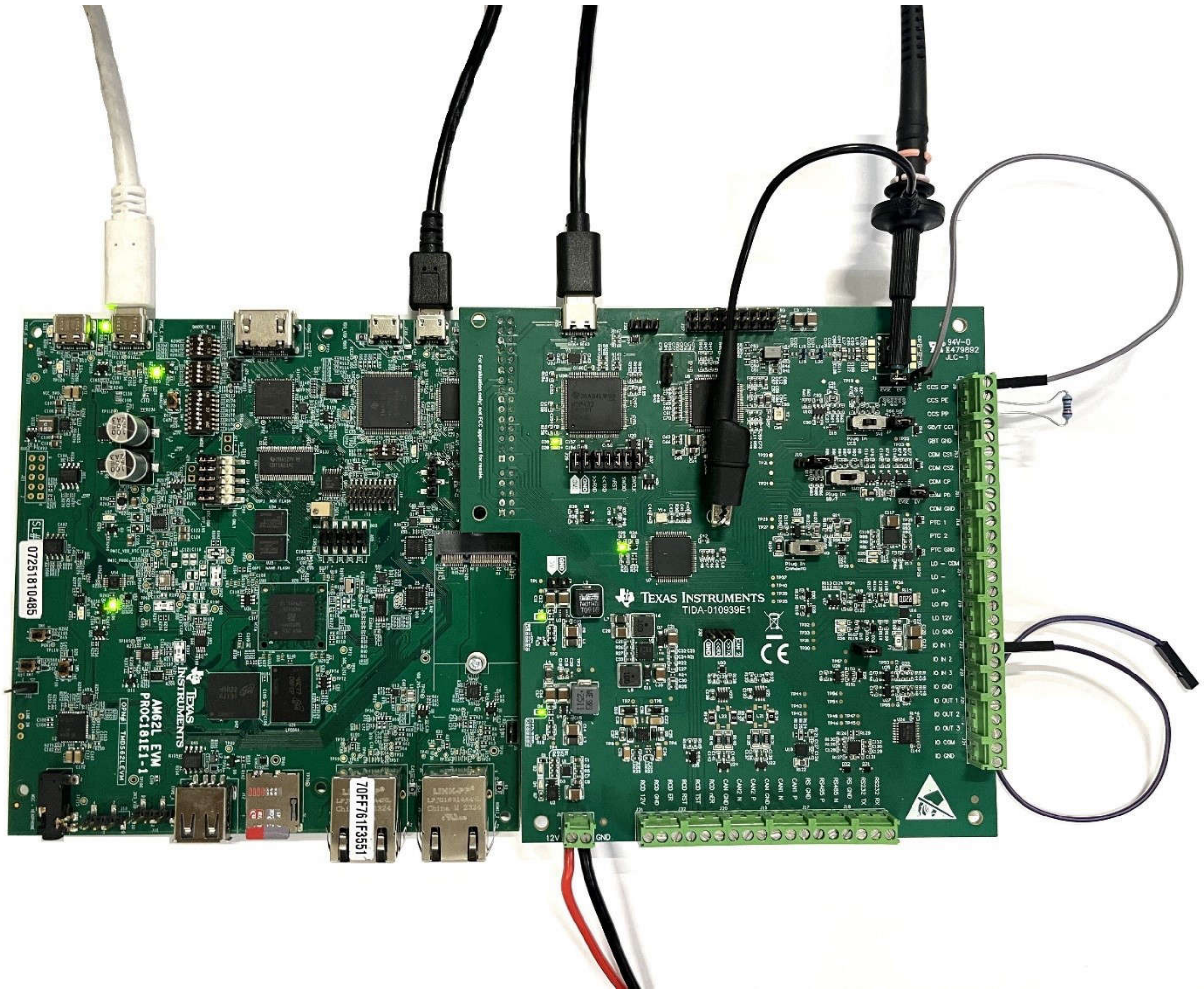


Figure 9-1. Hardware Setup

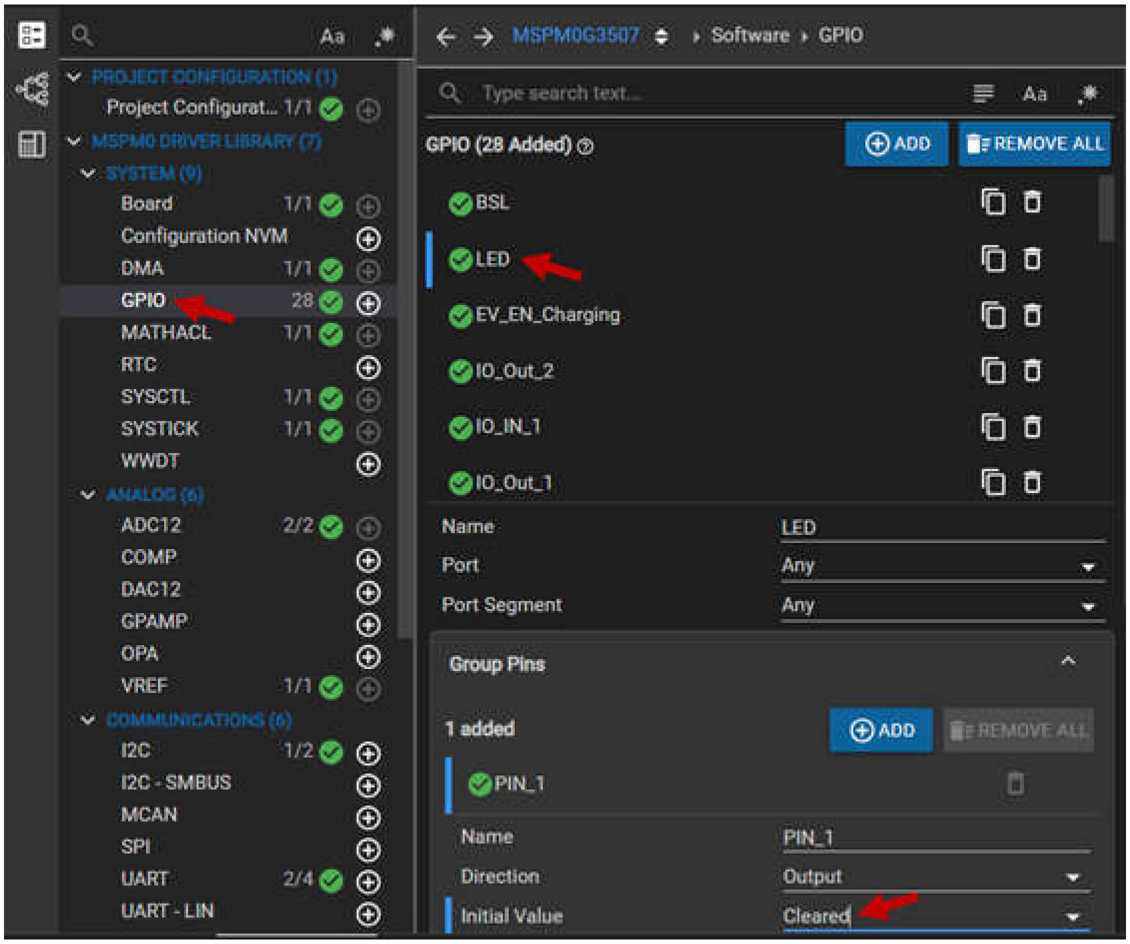
MSPM0 Software Setup

Open Code Composer Studio and select Browse software and examples and import the evse_controller_TIDA-010939 project located under Examples - LP-MSPM0G3507 - Demos. The example project is written to work together with the TIDA-010239 AC-Charger attached. Therefore, changes must be made to the code if the AM62L-EVSE-DEV-EVM is used standalone.

When power transfer is permitted, relays must close to connect the EV to the grid. In this example a LED will instead indicate when the power transfer is active. In the MSPM0 SDK software example, the LED is constantly active by default and must be changed accordingly:

- Open Code Composer Studio and select Browse software and examples
- Import the evse_controller_TIDA-010939 project located in the MSPM0 SDK under Examples / Demos.
- Open TIDA-010939.syscfg, open GPIO in the menu and select the GPIO named LED.

- Under Group Pins, the state of the pin after startup can be set as Initial Value. As this LED should only be active, when the power transfer is ongoing, the Initial Value is set to Cleared.



Name	LED
Port	Any
Port Segment	Any
Name	PIN_1
Direction	Output
Initial Value	Cleared

- After that close and save the Sysconfig and open the PowerSwitch.cpp located in the modules folder. The PowerSwitch.cpp handles the control of the high voltage relay. In the PowerSwitch.cpp you can find the functions PowerSwitch::switchOn() and PowerSwitch::switchOff().

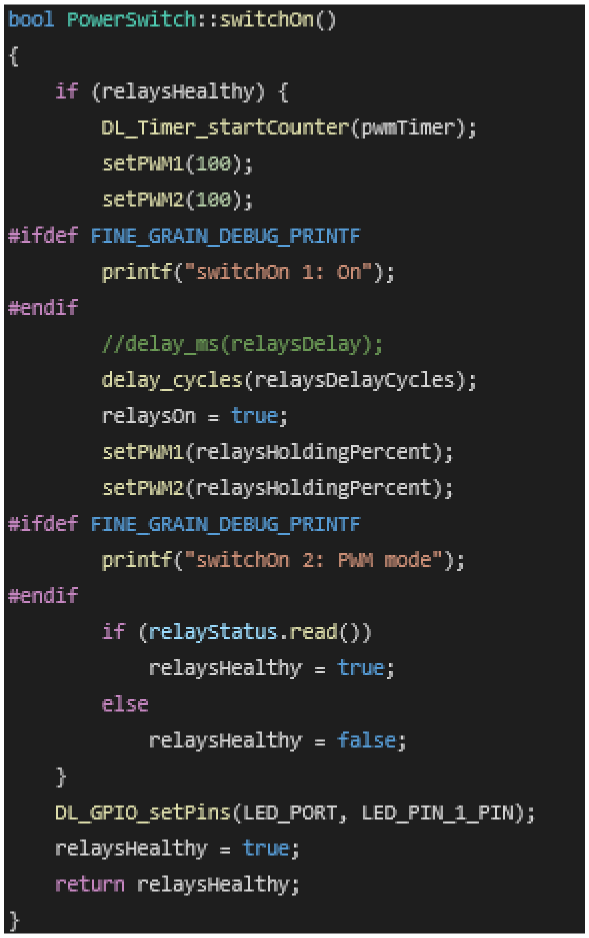
These functions will enable / disable a PWM as control signal for a relay driver. After switching the relay, the MSPM0 will read out a feedback signal, also called weld detection to verify the state of the relay. If switching of the relay was successful, so the contacts are not welded together, the variable relaysHealthy will be set to true. As the weld detection hardware is not available on the TIDA-010939, the relaysHealthy must be hardcoded to the code.

- Add to the end of the PowerSwitch::switchOn() function:

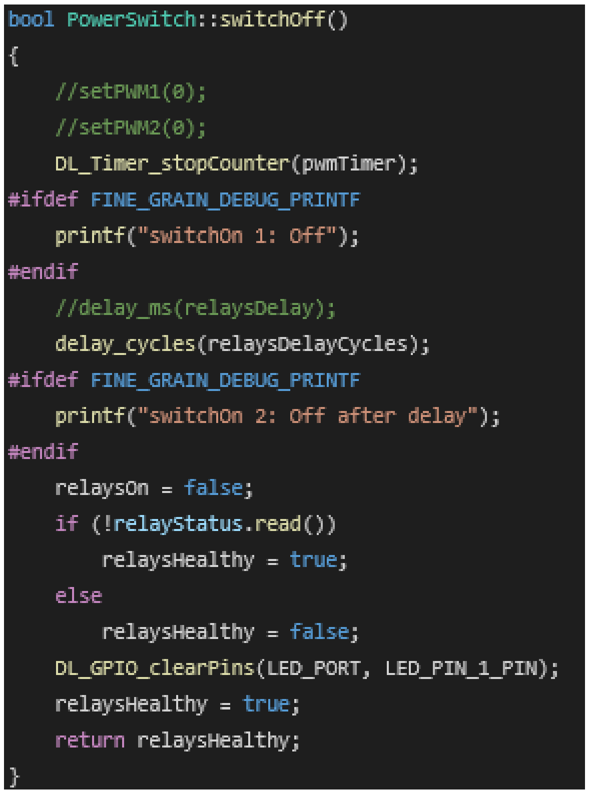
```
DL_GPIO_setPins(LED_PORT, LED_PIN_1_PIN);        // enables LED
relaysHealthy = true;                  // overwrites weld detection
```

- Add to the end of the PowerSwitch::switchOff() function:

```
DL_GPIO_clearPins(LED_PORT, LED_PIN_1_PIN);        // disables LED
relaysHealthy = true;                  // overwrites weld detection
```



```
bool PowerSwitch::switchOn()
{
    if (relaysHealthy) {
        DL_Timer_startCounter(pwmTimer);
        setPWM1(100);
        setPWM2(100);
#ifdef FINE_GRAIN_DEBUG_PRINTF
        printf("switchOn 1: On");
#endif
        //delay_ms(relaysDelay);
        delay_cycles(relaysDelayCycles);
        relaysOn = true;
        setPWM1(relaysHoldingPercent);
        setPWM2(relaysHoldingPercent);
#ifdef FINE_GRAIN_DEBUG_PRINTF
        printf("switchOn 2: PWM mode");
#endif
        if (relayStatus.read())
            relaysHealthy = true;
        else
            relaysHealthy = false;
    }
    DL_GPIO_setPins(LED_PORT, LED_PIN_1_PIN);
    relaysHealthy = true;
    return relaysHealthy;
}
```

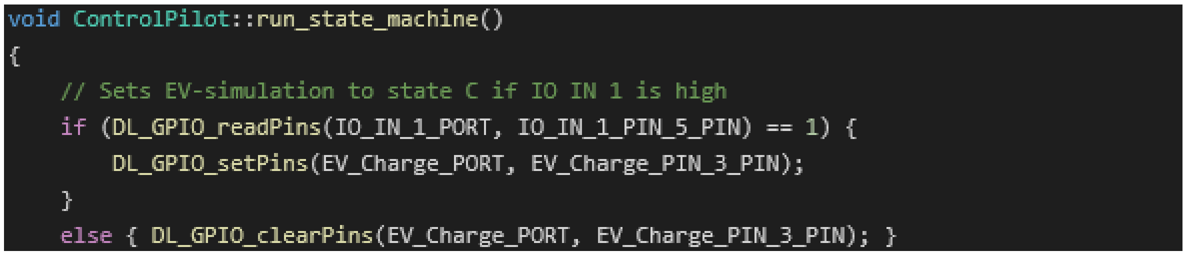


```
bool PowerSwitch::switchOff()
{
    //setPWM1(0);
    //setPWM2(0);
    DL_Timer_stopCounter(pwmTimer);
#ifdef FINE_GRAIN_DEBUG_PRINTF
    printf("switchOn 1: Off");
#endif
    //delay_ms(relaysDelay);
    delay_cycles(relaysDelayCycles);
#ifdef FINE_GRAIN_DEBUG_PRINTF
    printf("switchOn 2: Off after delay");
#endif
    relaysOn = false;
    if (!relayStatus.read())
        relaysHealthy = true;
    else
        relaysHealthy = false;
    DL_GPIO_clearPins(LED_PORT, LED_PIN_1_PIN);
    relaysHealthy = true;
    return relaysHealthy;
}
```

In this example the EV-simulation circuitry included on the AM62L-EVSE-DEV-EVM is used to simulate the behavior of an EV. Switch SW1 connects the EV – State B (9V), by placing a 2.74kΩ resistor to the Control Pilot signal. State C (6V) – request energy transfer can be enabled by connecting an additional 1.30kΩ resistor to the CP signal. This resistor is controlled by the MSPM0 through a MOSFET. In this test, State C should be manually controlled by the IO IN 1 of the AM62L-EVSE-DEV-EVM through a jumper wire connected to it. If the input is set to a logic high, State C is enabled to request power transfer. Therefore, the IO IN 1 pin must be read out.

- Add to the beginning of the ControlPilot::run_state_machine() function, located in ControlPilot.cpp:

```
if (DL_GPIO_readPins(IO_IN_1_PORT, IO_IN_1_PIN_5_PIN) == 1) {
        DL_GPIO_setPins(EV_Charge_PORT, EV_Charge_PIN_3_PIN);
}
else {DL_GPIO_clearPins(EV_Charge_PORT, EV_Charge_PIN_3_PIN);}
```



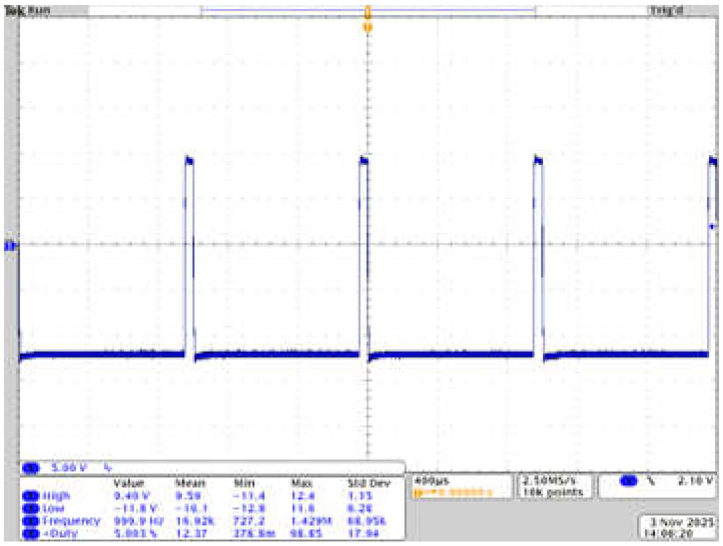
```
void ControlPilot::run_state_machine()
{
    // Sets EV-simulation to state C if IO IN 1 is high
    if (DL_GPIO_readPins(IO_IN_1_PORT, IO_IN_1_PIN_5_PIN) == 1) {
        DL_GPIO_setPins(EV_Charge_PORT, EV_Charge_PIN_3_PIN);
    }
    else { DL_GPIO_clearPins(EV_Charge_PORT, EV_Charge_PIN_3_PIN); }
```

Now the state of the EV can be changed, before the Control Pilot is measured by the EVSE. The CP is measured after the RCD, Lock, Proximity Pilot and permission to enable the power transfer are checked. The state of the Control Pilot is set by the ControlPilot::read_from_car function later in the ControlPilot.cpp.

- Compile and flash the code on the MSPM0.

Simulating an EV Charging Session with EVerest on AM62L

- System Startup
  - Connect the AM62L EVM to a USB-C power supply.
  - Open a serial terminal (COM port) on your PC. The default baud rate is 115200.
- After the system has booted, log in to the Linux shell.
- To monitor the EVerest logs in real time, run: journalctl -fu everest
- Attach an oscilloscope probe to the CCS Control Pilot (CP) signal (TP19)
  - The idle voltage level should be approximately +12 V
- In the EVerest log you will see messages such as:
  - Current Limit: 6
  - 	Ready to start charging
- Connect the EV simulation circuitry by moving switch 1 (SW1) to the Plug-IN position (toward Pin 1).
- The CP voltage now toggles between +9 V / −12 V, corresponding to State B – EV connected.
- The PWM duty cycle is 5 %, which signals a request for digital communication (HLC / ISO 15118). In this configuration EVerest initially attempts to start charging using ISO 15118 (HLC):
  - CP state changed: A -> B
  - EVSE IEC Session Started: EVConnected
  - EVSE IEC Set PWM On (5.0%) took 0 ms



	Value	Mean	Min	Max	Std Dev
High	9.40 V	9.59	−11.4	12.4	1.35
Low	−11.8 V	−10.1	−12.8	11.6	6.28
Frequency	999.9 Hz	16.92k	727.2	1.429M	68.95k
+Duty	5.003 %	12.37	276.5m	98.65	17.94

- EVerest waits for approximately 30 seconds for a PLC (Power Line Communication) response from the EV:
  - EVSE IEC AC mode, HLC enabled (ac_enforce_hlc), keeping 5 percent on until a dlink error is signalled
- If no valid HLC connection is established, the PWM is reset and a legacy IEC 61851-1 session is initiated:
  - EVSE IEC EV did not transition to state C, trying one legacy wakeup according to IEC61851-1 A.5.3
- After the second 30 sec. HLC timeout, EVerest starts a traditional IEC 61851-1 PWM-based charging session:
  - EVSE IEC Set PWM On (10%) took 0 ms
- Now the PWM is set to 10% which means the EV is allowed to draw up to 6A. This equals the maximum current capability of the charger, configured in this EVerest setup. Now the power transfer (State C) can be requested by connecting IO IN 1 to 5V (e.g. J2 Pin 2). The log will show the updated state and the start of the charging process:
  - CP state changed: B -> C
  - CAR IEC Event: CarRequestedPower
  - EVSE IEC Charger state: CarPaused -> Charging
- LED D13 on the AM62L-EVSE-DEV-EVM board will light up, indicating that the relays are closed and power transfer to the EV is active. If the jumper wire gets disconnected from IO IN 1, the charging session will pause until the EV returns to state C, or the charging session ends by disconnecting the EV from the charging station.

Changing the EVerest configuration

In the default setup, EVerest is configured to AC charge according to ISO15118 and allows a maximum charging current of 6 A on a single phase. The EVerest configuration is stored on the AM62L under:

/etc/everest/config.yaml

The configuration file config.yaml defines which modules are active, how they are connected, and what each module's parameters are. Each module corresponds to a logical function in the EVSE, such as authentication, metering, or communication.

Table 9-1. Modules and Functions

Module Name	Function
evse_manager	Main charge point logic: manages plug states, current limits, PWM generation, and session handling.
grid_connection_point	Defines grid limits such as fuse size and available phases. Acts as the upstream energy node.
tida_mcu	Hardware interface for the AM62L-EVSE-DEV-EVM board. Controls contactor, CP signal, RCD, lock, and reads inputs.
auth	Manages authorization of users (RFID, Plug & Charge, or external).
api	Provides a REST/WebSocket interface for monitoring and control. Used by dashboards or external systems.
energy_manager	Performs load balancing between multiple EVSEs or grid nodes.
persistent_store	Stores session and configuration data in a database (SQLite).
token_provider / token_validator	Used for testing or integrating external authentication backends (RFID, OCPP).
Slac	Handles HomePlug GreenPHY communication (mandatory for ISO 15118).

In this configuration, ISO15118 is represented by the EvseV2G module, in the iso15118_charger module:

iso15118_charger:
- module: EvseV2G
- config_module:
  - device: seth0 # PLC network interface
- connections:
  - security:
    - implementation_id: main
    - module_id: evse_security

Digital communication (HLC), is set in evse_manager:
- ac_hlc_enabled: true
- ac_enforce_hlc: true

If disabled, EVerest will fall back to analog IEC 61851-1 PWM-based charging.

To change the number of phases and the maximum current settings, the grid_connection_point and tida_mcu modules must be adjusted. The following example shows a 3-Phase, 16A configuration:
- Grid Connection Point: Defines the available grid capacity, important for load sharing:
  - grid_connection_point:
  - config_module:
    - fuse_limit_A: 16
    - phase_count: 3
- MCU Interface (TIDA Board): Defines hardware-level limits on current and number of phases:
  - tida_mcu:
  - module: TIDA010939BSP
  - config_module:
    - serial_port: /dev/ttyS2

- baud_rate: 115200
- min_current_A_import: 6
- max_current_A_import: 16
- min_phase_count_import: 3
- max_phase_count_import: 3
- has_socket: true

Setting it to 16 A changes the duty cycle to ≈ 27 %, corresponding to 16 A per phase. For more information on EVerest, please visit: everest.github.io.

10 Summary

The AM62L-EVSE-DEV-EVM User Guide provides comprehensive instructions for setting up and testing an Electric Vehicle Supply Equipment (EVSE) development platform that combines the AM62L processor with the AM62L-EVSE-DEV-EVM reference design [1]. The document begins with an introduction to the system architecture, which features the AM62L processor running the EVerest open-source charging stack for digital communications and the MSPM0G3507 microcontroller handling analog handshakes and safety functions.

The guide details the recommended test setup using two AM62L EVMs with AM62L-EVSE-DEV-EVM boards to simulate EVSE-to-EV communications without expensive test equipment. It covers software configuration including device tree overlays and explains how to test three key communication interfaces:

1. Power Line Communication (PLC) testing using the Lumissil Green PHY device for IP networking between EVSE and PEV
2. CAN bus communications for power conversion unit control
3. UART communications via RS-485 and RS-232 interfaces

Additionally, the document provides a detailed lab testing setup for the MSPM0 microcontroller, which simulates an AC charging session according to IEC 61851-1 standards. It explains how to modify the MSPM0 code in Code Composer Studio and configure the EVerest software stack to control charging parameters like maximum current and phase count.

The guide serves as a comprehensive resource for developers building EV charging solutions, enabling them to test and validate all communication interfaces required in modern EVSE systems across different charging standards and regional requirements.

11 Additional Information and Resources

- AM62L Product Page
- AM62L Product Overview
- AM62L Device Academy
- AM62L-EVSE-DEV-EVM Tool Folder
- AM62L-LINUX-SDK
- Electric Vehicle Supply Equipment Front-End Controller Reference Design (Rev. A)
- AM62L-EVSE-DEV-EVM User Guide
- MSPM0G3507 Product Page

STANDARD TERMS FOR EVALUATION MODULES

1. *Delivery:* TI delivers TI evaluation boards, kits, or modules, including any accompanying demonstration software, components, and/or documentation which may be provided together or separately (collectively, an “EVM” or “EVMs”) to the User (“User”) in accordance with the terms set forth herein. User's acceptance of the EVM is expressly subject to the following terms.

   1.1 EVMs are intended solely for product or software developers for use in a research and development setting to facilitate feasibility evaluation, experimentation, or scientific analysis of TI semiconductors products. EVMs have no direct function and are not finished products. EVMs shall not be directly or indirectly assembled as a part or subassembly in any finished product. For clarification, any software or software tools provided with the EVM (“Software”) shall not be subject to the terms and conditions set forth herein but rather shall be subject to the applicable terms that accompany such Software

   1.2 EVMs are not intended for consumer or household use. EVMs may not be sold, sublicensed, leased, rented, loaned, assigned, or otherwise distributed for commercial purposes by Users, in whole or in part, or used in any finished product or production system.

2 *Limited Warranty and Related Remedies/Disclaimers*:

   2.1 These terms do not apply to Software. The warranty, if any, for Software is covered in the applicable Software License Agreement.

   2.2 TI warrants that the TI EVM will conform to TI's published specifications for ninety (90) days after the date TI delivers such EVM to User. Notwithstanding the foregoing, TI shall not be liable for a nonconforming EVM if (a) the nonconformity was caused by neglect, misuse or mistreatment by an entity other than TI, including improper installation or testing, or for any EVMs that have been altered or modified in any way by an entity other than TI, (b) the nonconformity resulted from User's design, specifications or instructions for such EVMs or improper system design, or (c) User has not paid on time. Testing and other quality control techniques are used to the extent TI deems necessary. TI does not test all parameters of each EVM. User's claims against TI under this Section 2 are void if User fails to notify TI of any apparent defects in the EVMs within ten (10) business days after delivery, or of any hidden defects with ten (10) business days after the defect has been detected.

   2.3 TI's sole liability shall be at its option to repair or replace EVMs that fail to conform to the warranty set forth above, or credit User's account for such EVM. TI's liability under this warranty shall be limited to EVMs that are returned during the warranty period to the address designated by TI and that are determined by TI not to conform to such warranty. If TI elects to repair or replace such EVM, TI shall have a reasonable time to repair such EVM or provide replacements. Repaired EVMs shall be warranted for the remainder of the original warranty period. Replaced EVMs shall be warranted for a new full ninety (90) day warranty period.

WARNING

Evaluation Kits are intended solely for use by technically qualified, professional electronics experts who are familiar with the dangers and application risks associated with handling electrical mechanical components, systems, and subsystems.

User shall operate the Evaluation Kit within TI’s recommended guidelines and any applicable legal or environmental requirements as well as reasonable and customary safeguards. Failure to set up and/or operate the Evaluation Kit within TI’s recommended guidelines may result in personal injury or death or property damage. Proper set up entails following TI’s instructions for electrical ratings of interface circuits such as input, output and electrical loads.

NOTE:

EXPOSURE TO ELECTROSTATIC DISCHARGE (ESD) MAY CAUSE DEGREDATION OR FAILURE OF THE EVALUATION KIT; TI RECOMMENDS STORAGE OF THE EVALUATION KIT IN A PROTECTIVE ESD BAG.

3 *Regulatory Notices:*

3.1 *United States*

3.1.1 *Notice applicable to EVMs not FCC-Approved:*

FCC NOTICE: This kit is designed to allow product developers to evaluate electronic components, circuitry, or software associated with the kit to determine whether to incorporate such items in a finished product and software developers to write software applications for use with the end product. This kit is not a finished product and when assembled may not be resold or otherwise marketed unless all required FCC equipment authorizations are first obtained. Operation is subject to the condition that this product not cause harmful interference to licensed radio stations and that this product accept harmful interference. Unless the assembled kit is designed to operate under part 15, part 18 or part 95 of this chapter, the operator of the kit must operate under the authority of an FCC license holder or must secure an experimental authorization under part 5 of this chapter.

3.1.2 *For EVMs annotated as FCC – FEDERAL COMMUNICATIONS COMMISSION Part 15 Compliant:*

CAUTION

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

FCC Interference Statement for Class A EVM devices

NOTE: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

FCC Interference Statement for Class B EVM devices

NOTE: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- *Reorient or relocate the receiving antenna.*
- *Increase the separation between the equipment and receiver.*
- *Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.*
- *Consult the dealer or an experienced radio/TV technician for help.*

3.2 *Canada*

3.2.1 *For EVMs issued with an Industry Canada Certificate of Conformance to RSS-210 or RSS-247*

Concerning EVMs Including Radio Transmitters:

This device complies with Industry Canada license-exempt RSSs. Operation is subject to the following two conditions:

(1) this device may not cause interference, and (2) this device must accept any interference, including interference that may cause undesired operation of the device.

Concernant les EVMs avec appareils radio:

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes: (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

Concerning EVMs Including Detachable Antennas:

Under Industry Canada regulations, this radio transmitter may only operate using an antenna of a type and maximum (or lesser) gain approved for the transmitter by Industry Canada. To reduce potential radio interference to other users, the antenna type and its gain should be so chosen that the equivalent isotropically radiated power (e.i.r.p.) is not more than that necessary for successful communication. This radio transmitter has been approved by Industry Canada to operate with the antenna types listed in the user guide with the maximum permissible gain and required antenna impedance for each antenna type indicated. Antenna types not included in this list, having a gain greater than the maximum gain indicated for that type, are strictly prohibited for use with this device.

Concernant les EVMs avec antennes détachables

Conformément à la réglementation d'Industrie Canada, le présent émetteur radio peut fonctionner avec une antenne d'un type et d'un gain maximal (ou inférieur) approuvé pour l'émetteur par Industrie Canada. Dans le but de réduire les risques de brouillage radioélectrique à l'intention des autres utilisateurs, il faut choisir le type d'antenne et son gain de sorte que la puissance isotrope rayonnée équivalente (p.i.r.e.) ne dépasse pas l'intensité nécessaire à l'établissement d'une communication satisfaisante. Le présent émetteur radio a été approuvé par Industrie Canada pour fonctionner avec les types d'antenne énumérés dans le manuel d'usage et ayant un gain admissible maximal et l'impédance requise pour chaque type d'antenne. Les types d'antenne non inclus dans cette liste, ou dont le gain est supérieur au gain maximal indiqué, sont strictement interdits pour l'exploitation de l'émetteur

3.3 *Japan*

3.3.1 *Notice for EVMs delivered in Japan:* Please see http://www.tij.co.jp/lsds/ti_ja/general/eStore/notice_01.page 日本国内に輸入される評価用キット、ボードについては、次のところをご覧ください。

https://www.ti.com/ja-jp/legal/notice-for-evaluation-kits-delivered-in-japan.html

3.3.2 *Notice for Users of EVMs Considered "Radio Frequency Products" in Japan:* EVMs entering Japan may not be certified by TI as conforming to Technical Regulations of Radio Law of Japan.

If User uses EVMs in Japan, not certified to Technical Regulations of Radio Law of Japan, User is required to follow the instructions set forth by Radio Law of Japan, which includes, but is not limited to, the instructions below with respect to EVMs (which for the avoidance of doubt are stated strictly for convenience and should be verified by User):

1. Use EVMs in a shielded room or any other test facility as defined in the notification #173 issued by Ministry of Internal Affairs and Communications on March 28, 2006, based on Sub-section 1.1 of Article 6 of the Ministry's Rule for Enforcement of Radio Law of Japan,
2. Use EVMs only after User obtains the license of Test Radio Station as provided in Radio Law of Japan with respect to EVMs, or
3. Use of EVMs only after User obtains the Technical Regulations Conformity Certification as provided in Radio Law of Japan with respect to EVMs. Also, do not transfer EVMs, unless User gives the same notice above to the transferee. Please note that if User does not follow the instructions above, User will be subject to penalties of Radio Law of Japan.

【無線電波を送信する製品の開発キットをお使いになる際の注意事項】 開発キットの中には技術基準適合証明を受けていないものがあります。 技術適合証明を受けていないもののご使用に際しては、電波法遵守のため、以下のいずれかの措置を取っていただく必要がありますのでご注意ください。

1. 電波法施行規則第6条第1項第1号に基づく平成18年3月28日総務省告示第173号で定められた電波暗室等の試験設備でご使用いただく。
2. 実験局の免許を取得後ご使用いただく。
3. 技術基準適合証明を取得後ご使用いただく。

なお、本製品は、上記の「ご使用にあたっての注意」を譲渡先、移転先に通知しない限り、譲渡、移転できないものとします。

上記を遵守頂けない場合は、電波法の罰則が適用される可能性があることをご留意ください。 日本テキサス･ インスツルメンツ株式会社

東京都新宿区西新宿６丁目２４番１号

西新宿三井ビル

3.3.3 *Notice for EVMs for Power Line Communication:* Please see http://www.tij.co.jp/lsds/ti_ja/general/eStore/notice_02.page

電力線搬送波通信についての開発キットをお使いになる際の注意事項については、次のところをご覧ください。https://www.ti.com/ja-jp/legal/notice-for-evaluation-kits-for-power-line-communication.html

3.4 *European Union*

3.4.1 *For EVMs subject to EU Directive 2014/30/EU (Electromagnetic Compatibility Directive)*:

This is a class A product intended for use in environments other than domestic environments that are connected to a low-voltage power-supply network that supplies buildings used for domestic purposes. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

4 *EVM Use Restrictions and Warnings:*

4.1 EVMS ARE NOT FOR USE IN FUNCTIONAL SAFETY AND/OR SAFETY CRITICAL EVALUATIONS, INCLUDING BUT NOT LIMITED TO EVALUATIONS OF LIFE SUPPORT APPLICATIONS.

4.2 User must read and apply the user guide and other available documentation provided by TI regarding the EVM prior to handling or using the EVM, including without limitation any warning or restriction notices. The notices contain important safety information related to, for example, temperatures and voltages.

4.3 *Safety-Related Warnings and Restrictions:*

4.3.1 User shall operate the EVM within TI's recommended specifications and environmental considerations stated in the user guide, other available documentation provided by TI, and any other applicable requirements and employ reasonable and customary safeguards. Exceeding the specified performance ratings and specifications (including but not limited to input and output voltage, current, power, and environmental ranges) for the EVM may cause personal injury or death, or property damage. If there are questions concerning performance ratings and specifications, User should contact a TI field representative prior to connecting interface electronics including input power and intended loads. Any loads applied outside of the specified output range may also result in unintended and/or inaccurate operation and/or possible permanent damage to the EVM and/or interface electronics. Please consult the EVM user guide prior to connecting any load to the EVM output. If there is uncertainty as to the load specification, please contact a TI field representative. During normal operation, even with the inputs and outputs kept within the specified allowable ranges, some circuit components may have elevated case temperatures. These components include but are not limited to linear regulators, switching transistors, pass transistors, current sense resistors, and heat sinks, which can be identified using the information in the associated documentation. When working with the EVM, please be aware that the EVM may become very warm.

4.3.2 EVMs are intended solely for use by technically qualified, professional electronics experts who are familiar with the dangers and application risks associated with handling electrical mechanical components, systems, and subsystems. User assumes all responsibility and liability for proper and safe handling and use of the EVM by User or its employees, affiliates, contractors or designees. User assumes all responsibility and liability to ensure that any interfaces (electronic and/or mechanical) between the EVM and any human body are designed with suitable isolation and means to safely limit accessible leakage currents to minimize the risk of electrical shock hazard. User assumes all responsibility and liability for any improper or unsafe handling or use of the EVM by User or its employees, affiliates, contractors or designees.

4.4 User assumes all responsibility and liability to determine whether the EVM is subject to any applicable international, federal, state, or local laws and regulations related to User's handling and use of the EVM and, if applicable, User assumes all responsibility and liability for compliance in all respects with such laws and regulations. User assumes all responsibility and liability for proper disposal and recycling of the EVM consistent with all applicable international, federal, state, and local requirements.

5. *Accuracy of Information:* To the extent TI provides information on the availability and function of EVMs, TI attempts to be as accurate as possible. However, TI does not warrant the accuracy of EVM descriptions, EVM availability or other information on its websites as accurate, complete, reliable, current, or error-free.

6. *Disclaimers:*

6.1 EXCEPT AS SET FORTH ABOVE, EVMS AND ANY MATERIALS PROVIDED WITH THE EVM (INCLUDING, BUT NOT LIMITED TO, REFERENCE DESIGNS AND THE DESIGN OF THE EVM ITSELF) ARE PROVIDED "AS IS" AND "WITH ALL FAULTS." TI DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING SUCH ITEMS, INCLUDING BUT NOT LIMITED TO ANY EPIDEMIC FAILURE WARRANTY OR IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADE SECRETS OR OTHER INTELLECTUAL PROPERTY RIGHTS.

6.2 EXCEPT FOR THE LIMITED RIGHT TO USE THE EVM SET FORTH HEREIN, NOTHING IN THESE TERMS SHALL BE CONSTRUED AS GRANTING OR CONFERRING ANY RIGHTS BY LICENSE, PATENT, OR ANY OTHER INDUSTRIAL OR INTELLECTUAL PROPERTY RIGHT OF TI, ITS SUPPLIERS/LICENSORS OR ANY OTHER THIRD PARTY, TO USE THE EVM IN ANY FINISHED END-USER OR READY-TO-USE FINAL PRODUCT, OR FOR ANY INVENTION, DISCOVERY OR IMPROVEMENT, REGARDLESS OF WHEN MADE, CONCEIVED OR ACQUIRED.

7. *USER'S INDEMNITY OBLIGATIONS AND REPRESENTATIONS.* USER WILL DEFEND, INDEMNIFY AND HOLD TI, ITS LICENSORS AND THEIR REPRESENTATIVES HARMLESS FROM AND AGAINST ANY AND ALL CLAIMS, DAMAGES, LOSSES, EXPENSES, COSTS AND LIABILITIES (COLLECTIVELY, "CLAIMS") ARISING OUT OF OR IN CONNECTION WITH ANY HANDLING OR USE OF THE EVM THAT IS NOT IN ACCORDANCE WITH THESE TERMS. THIS OBLIGATION SHALL APPLY WHETHER CLAIMS ARISE UNDER STATUTE, REGULATION, OR THE LAW OF TORT, CONTRACT OR ANY OTHER LEGAL THEORY, AND EVEN IF THE EVM FAILS TO PERFORM AS DESCRIBED OR EXPECTED.

8. *Limitations on Damages and Liability:*

   8.1 *General Limitations*. IN NO EVENT SHALL TI BE LIABLE FOR ANY SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL, OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF THESE TERMS OR THE USE OF THE EVMS , REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. EXCLUDED DAMAGES INCLUDE, BUT ARE NOT LIMITED TO, COST OF REMOVAL OR REINSTALLATION, ANCILLARY COSTS TO THE PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, RETESTING, OUTSIDE COMPUTER TIME, LABOR COSTS, LOSS OF GOODWILL, LOSS OF PROFITS, LOSS OF SAVINGS, LOSS OF USE, LOSS OF DATA, OR BUSINESS INTERRUPTION. NO CLAIM, SUIT OR ACTION SHALL BE BROUGHT AGAINST TI MORE THAN TWELVE (12) MONTHS AFTER THE EVENT THAT GAVE RISE TO THE CAUSE OF ACTION HAS OCCURRED.

   8.2 *Specific Limitations.* IN NO EVENT SHALL TI'S AGGREGATE LIABILITY FROM ANY USE OF AN EVM PROVIDED HEREUNDER, INCLUDING FROM ANY WARRANTY, INDEMITY OR OTHER OBLIGATION ARISING OUT OF OR IN CONNECTION WITH THESE TERMS, , EXCEED THE TOTAL AMOUNT PAID TO TI BY USER FOR THE PARTICULAR EVM(S) AT ISSUE DURING THE PRIOR TWELVE (12) MONTHS WITH RESPECT TO WHICH LOSSES OR DAMAGES ARE CLAIMED. THE EXISTENCE OF MORE THAN ONE CLAIM SHALL NOT ENLARGE OR EXTEND THIS LIMIT.

9. *Return Policy.* Except as otherwise provided, TI does not offer any refunds, returns, or exchanges. Furthermore, no return of EVM(s) will be accepted if the package has been opened and no return of the EVM(s) will be accepted if they are damaged or otherwise not in a resalable condition. If User feels it has been incorrectly charged for the EVM(s) it ordered or that delivery violates the applicable order, User should contact TI. All refunds will be made in full within thirty (30) working days from the return of the components(s), excluding any postage or packaging costs.

10. *Governing Law:* These terms and conditions shall be governed by and interpreted in accordance with the laws of the State of Texas, without reference to conflict-of-laws principles. User agrees that non-exclusive jurisdiction for any dispute arising out of or relating to these terms and conditions lies within courts located in the State of Texas and consents to venue in Dallas County, Texas. Notwithstanding the foregoing, any judgment may be enforced in any United States or foreign court, and TI may seek injunctive relief in any United States or foreign court.