# S32V234 Customer EVB Setup Guide for Vision SDK


| ABSTRACT: |
|---|
| The document describes the installation and setup of the S32V234 Customer Evaluation Board for use with Vision SDK. |



| KEYWORDS: |
|---|
| S32V234, EVB, Customer EVB, Setup |
| APPROVED: |

# Revision History

| VERSION | DATE | AUTHOR | CHANGE DESCRIPTION |
|---|---|---|---|
| 0.5 | 13-July-2015 | Rostislav Hulik | First draft |
| 0.6 | 20-August-2015 | Rostislav Hulik | Linux OS support added |
| 0.7 | 7-December-2015 | Tomas Babinec | Linux OS support updated |
| 0.8 | 28-January-2016 | Tomas Babinec | Linux OS support updated |
| 0.9 | 6-May-2016 | Rostislav Hulik | Updated to REV D EVB |
| 1.0 | 9-March-2017 | Tomas Babinec | Security fuse instructions added<br>Moved to NXP document template |
| 1.1 | 17-May-2017 | Rostislav Hulik | Added Lauterbach instruction for Linux host |
| 1.2 | 22-August-2018 | Khang Ba Tran | Updated for VSDK RTM 1.2.0 release |

# Table of Contents

# 1 Introduction

In this document, the S32V234 Customer EVB setup is described. The SDK supports Standalone (OS less) and Linux runtime environment.

The document covers installation of the hardware (REV C EVB). This includes necessary prerequisites and setup of the S32V234 evaluation board.

## 1.1 Purpose

The purpose of this document is to guide the user along the installation procedure of the S32V234 EVB.

## 1.2 Audience Description

This document is intended to S32V234 EVB Vision SDK users.

## 1.3 References

| *Id* | *Title* | *Location/Version* |
|---|---|---|
| *[1]* | *S32V234 Customer evaluation board* | *S32V234-EVB REV C* |

**Table 1 References Table**

## 1.4 Definitions, Acronyms, and Abbreviations

| *Term/Acronym* | *Description* |
|---|---|
| *ACF* | *APEX Core Framework* |
| *ARM* | *Family of RISC architectures* |
| *S32V234* | *S32V234 SoC* |
| *SDK* | *System Development Kit* |

**Table 2 Acronyms Table**

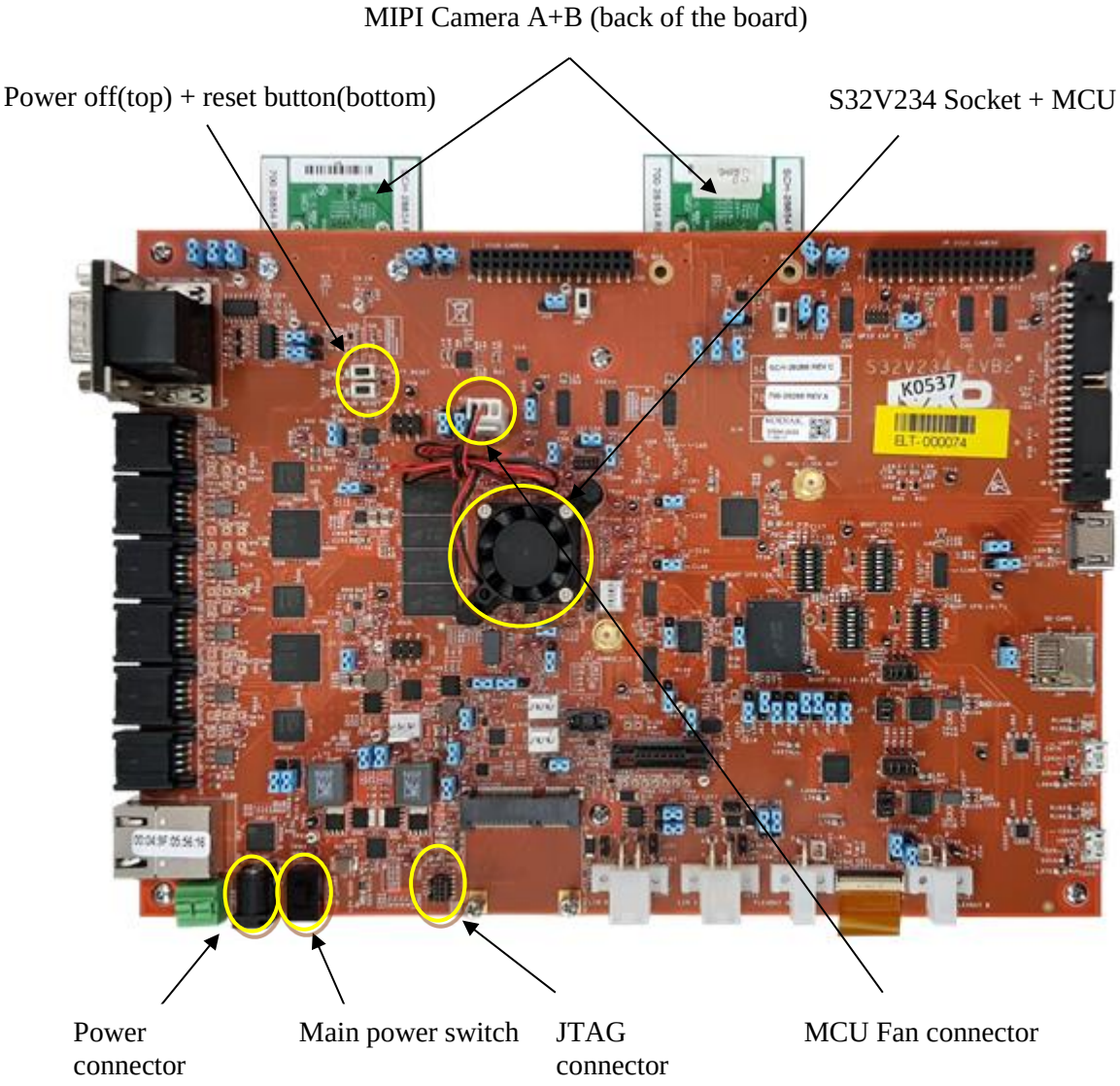
## 1.5 Document Location

This document is available in VisionSDK directory structure at the following location:
s32v234_sdk/docs/vsdk/S32V234-EVB_SetupGuide.pdf

# 2 Hardware Overview

## 2.1 S32V234 EVB

On **Figure 1**, the S32V234 Evaluation board is displayed. Please note the positions of all necessary jumpers and vital parts on board:



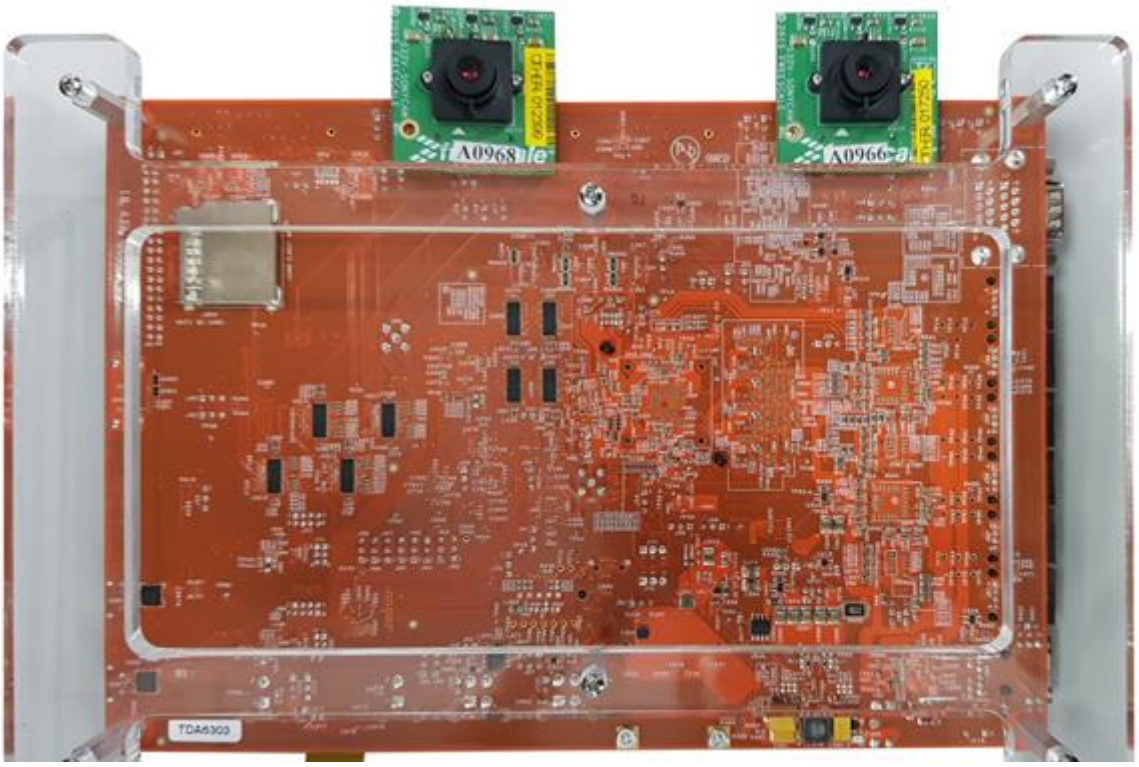


**Figure 1** S32V234 EVB (front and back side)

## 2.2 S32V234 EVB HW Setup

The setup of the board for Vision SDK consists of following steps:

1. Ensure the power switch is in top position (OFF)
2. Connect the power cable to the board (Figure 2)
3. (Optional) Connect the Lauterbach debugger to the JTAG Connector (Figure 2)
4. (Optional) Connect the MCU Fan connectors (Figure 1) (Fan is not necessary when not using GPU on S32V234)
5. (Optional) Disable security fuse (important to run ISP subsystem related demos)
    a. Execute the s32_sec_fuse.cmm Lauterbach script from s32v234_sdk/os/debug/lauterbach/security_fuse directory.
    b. The script executes ocotp_m4_sram.elf binary on M4 core and disables security fuse to enable access to KRAM from A53 cores.

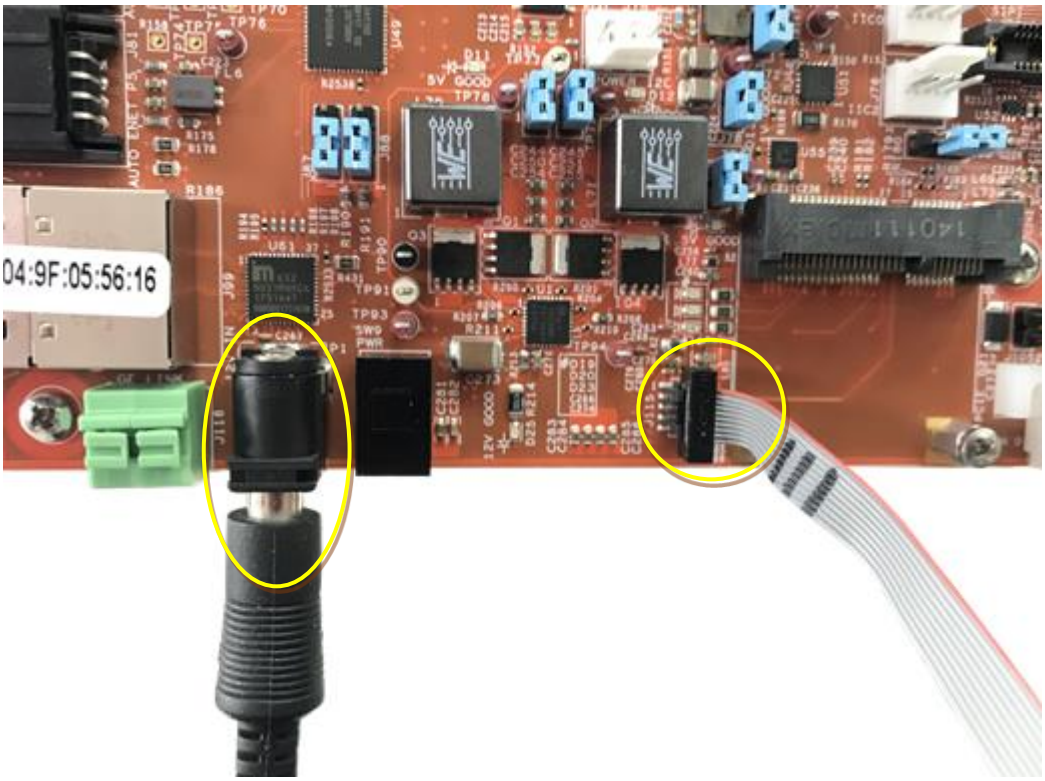



**Figure 2** Power and JTAG connections

# 3 SDK Applications startup

## 3.1 Running the standalone applications via SD card boot

The build applications for OSless environment are possible to run directly from the SD card.

1. Build the application in `s32v234/demos/*/build-v234ce-gnu-sa-*` directory – the name_of_demo.bin file is produces
2. Prepare the SD card (Micro SDHC USH-I card is recommended):
    a. Linux – use dd command for loading the compiled .bin file

    ```
    sudo dd if=demo.bin of=/dev/sdb bs=512 seek=0 conv=fsync
    ```

    (the SD card is in /dev/sdb in this case).

    b. Windows – use Win32 Disk Imager utility for loading the name_of_demo.bin file into SD card
3. Insert the Micro SD card into main SD card slot and turn on the board.

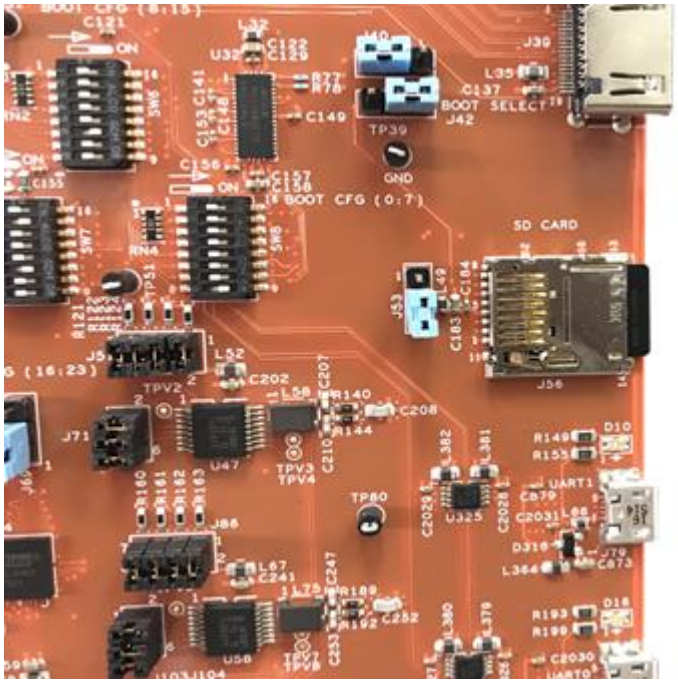



**Figure 3** SD Card placement

## 3.2 Running the standalone applications via Lauterbach debugger

After successful connection of the hardware, it's possible to run the standalone applications via Lauterbach debugger.

### 3.2.1 Lauterbach T32 setup using setup script (Linux host)

1. Necessary prerequisites:
    a. Environment variable `S32V234_SDK_ROOT` pointing to the s32v234_sdk directory (should be set by the installer)
    b. Lauterbach Trace32 v2.3.3 (tested on S.2015.03.00061311)
2. Execute the `s32v234_sdk/os/debug/lauterbach/LINUX_S32V234.sh` configuration file.
3. Two T32 instances will open, from now, only PowerView for CA53 will be used (all scripts are ran from here) (Figure 4)

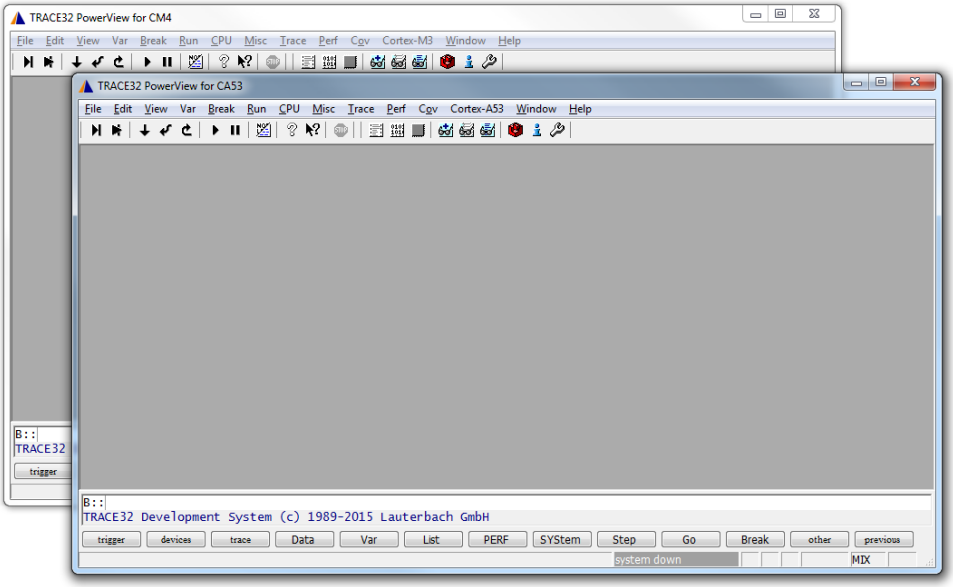



**Figure 4** PowerView windows for S32V234 debug

4. Before every run of the demo/application, the HW reset needs to be issued – press the top and bottom reset button (Figure 1) in this exact order.
5. Run the script in `s32v234/demos/*/build-v234ce-gnu-sa-*/*.cmm` to run the specific demo (Figure 5)
    a. If any warning comes during the load of the application ("Debugger acts like a slave…"), please close all PowerView windows and repeat from the point 3.

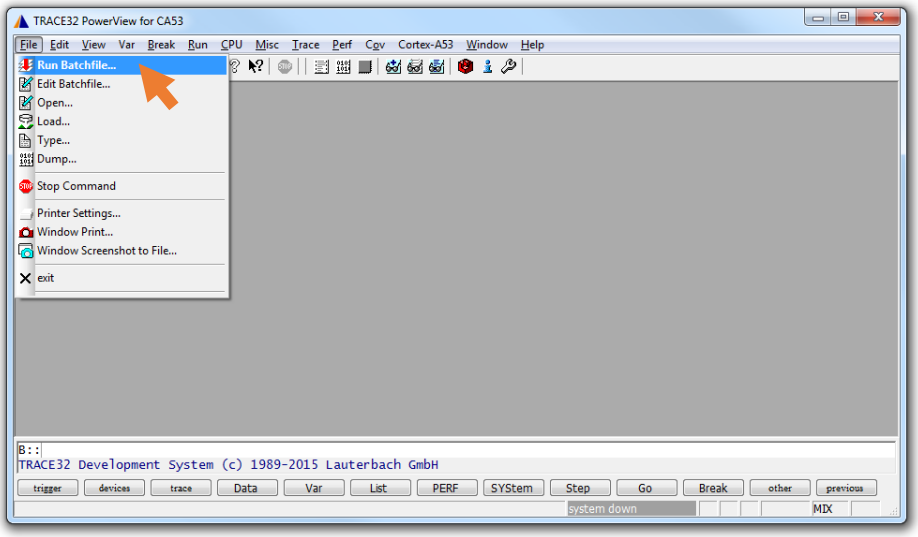



**Figure 5** Running the script for application start

6. For every other reset, please repeat steps 5. and 6.

## 3.2.2Lauterbach T32Start setup using setup script (Windows host)

7. Necessary prerequisites:
    a. Environment variable `S32V234_SDK_ROOT` pointing to the s32v234_sdk directory (should be set by the installer)
    b. Lauterbach Trace32 v2.3.3 (tested on S.2015.03.00061311)
8. Open the `s32v234_sdk/os/debug/lauterbach/S32V234.ts2` configuration file
    a. (optional) Or open the T32Start, right click to the blank area, choose "File->Load from file and add" and select this file (Figure 6)

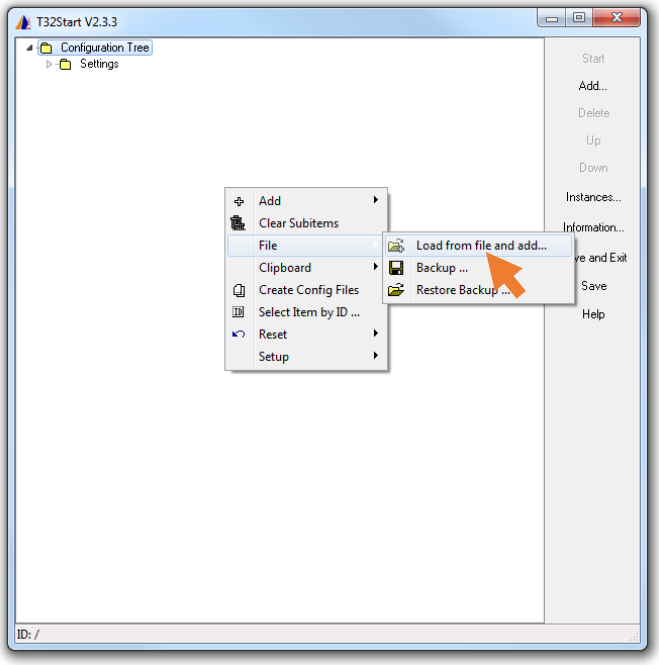



**Figure 6** T32Start configuration setup

9. On opened T32Start window, select S32V234 EVB setting and press "Start" (Figure 7)

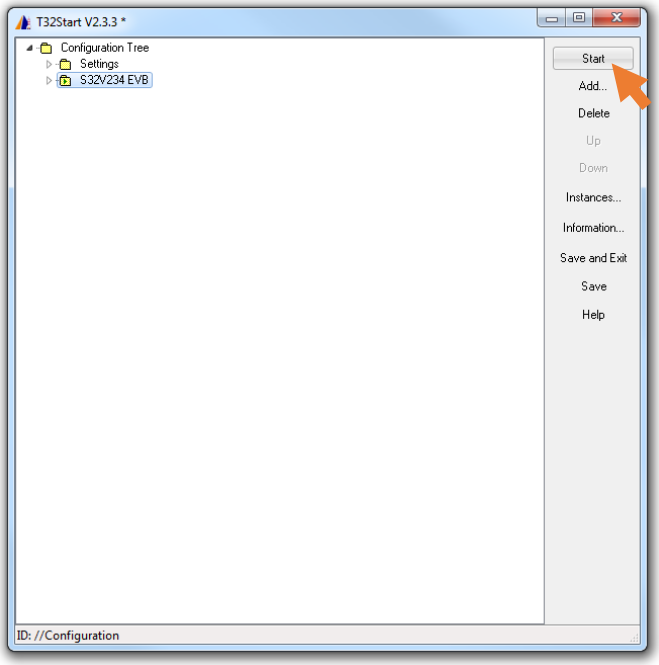



**Figure 7** Starting the configuration

10. Two T32 instances will open, from now, only PowerView for CA53 will be used (all scripts are ran from here) (Figure 8)

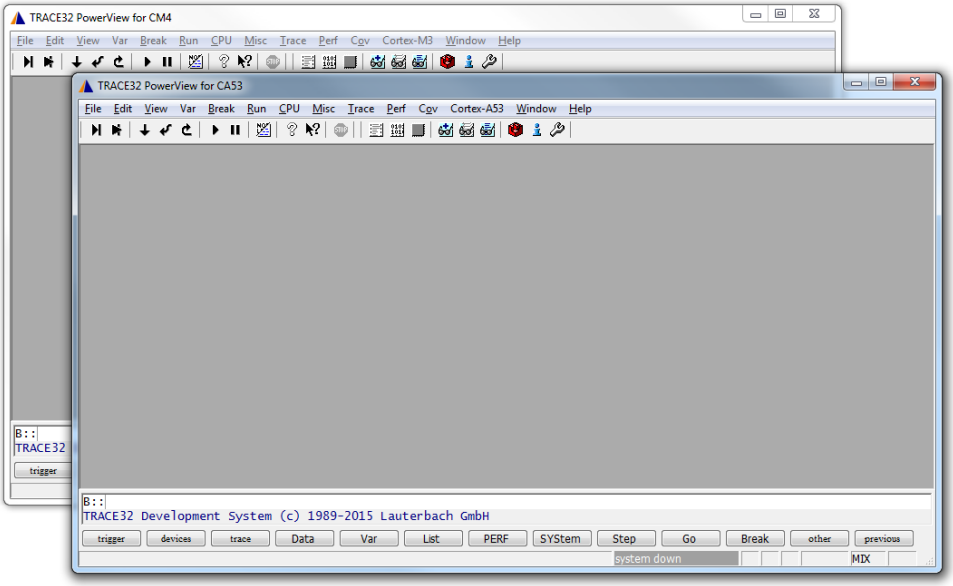



**Figure 8** PowerView windows for S32V234 debug

11. Before every run of the demo/application, the HW reset needs to be issued – press the top and bottom reset button (Figure 1) in this exact order.

12. Run the script in `s32v234/demos/*/build-v234ce-gnu-sa-*/*.cmm` to run the specific demo (Figure 9)
    a. If any warning comes during the load of the application ("Debugger acts like a slave…"), please close all PowerView windows and repeat from the point 3.

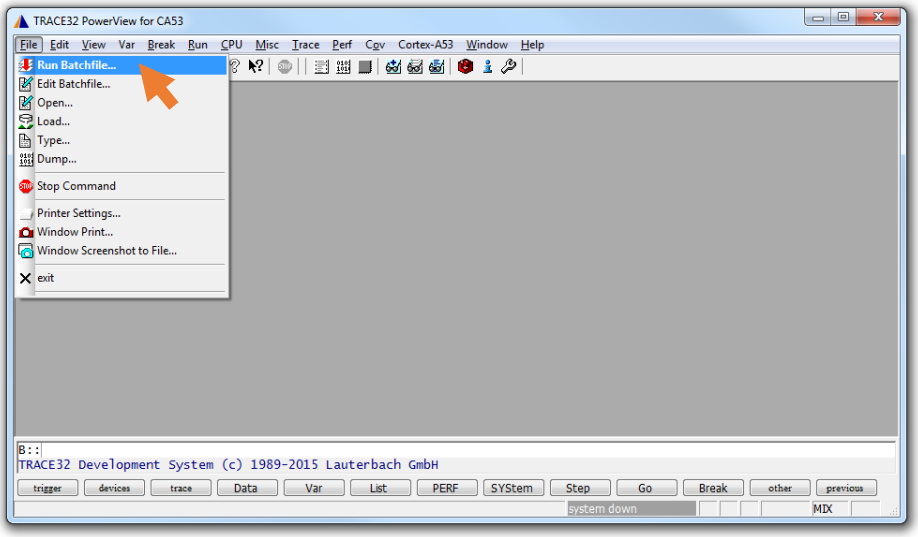


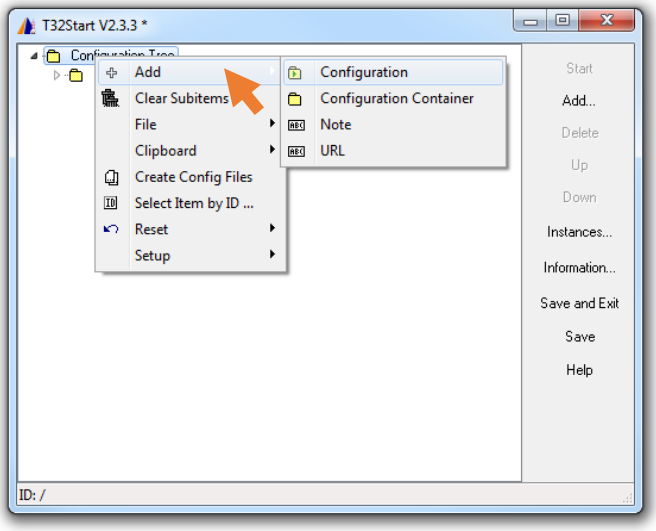

**Figure 9** Running the script for application start

13. For every other reset, please repeat steps 5. and 6.

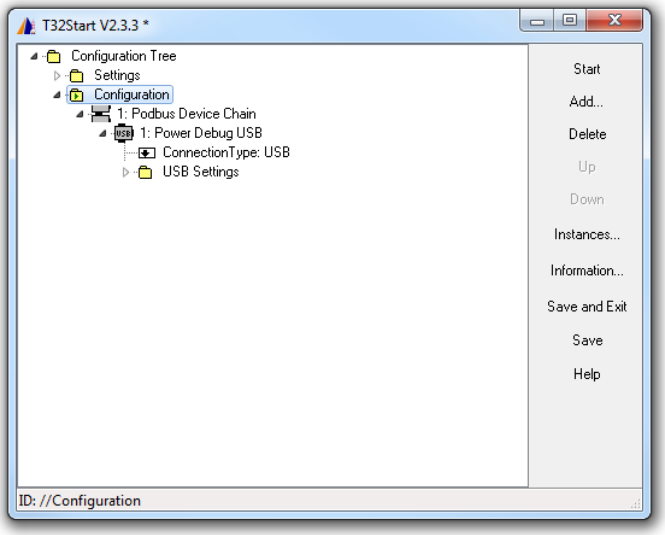
## 3.2.3Manual Lauterbach T32Start setup (Windows host)

If the setup script is not working or there is a need for specific T32 setup, following procedure will enable user to set the configuration manually:

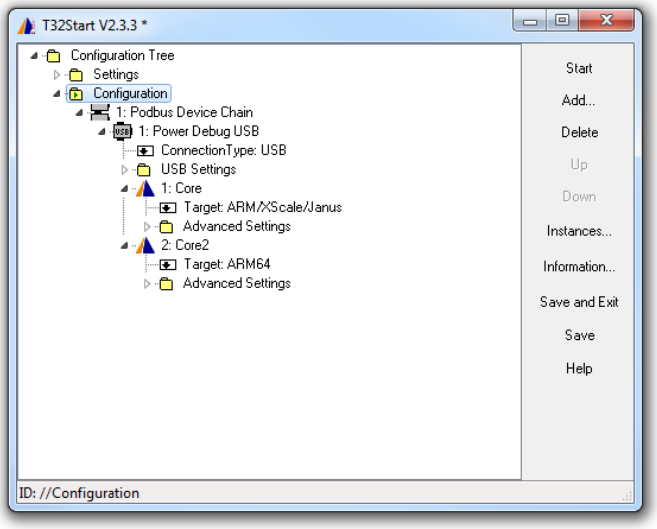
1. Open the T32Start window and add a new configuration by right-clicking the Configuration Tree (It can be renamed arbitrarily).



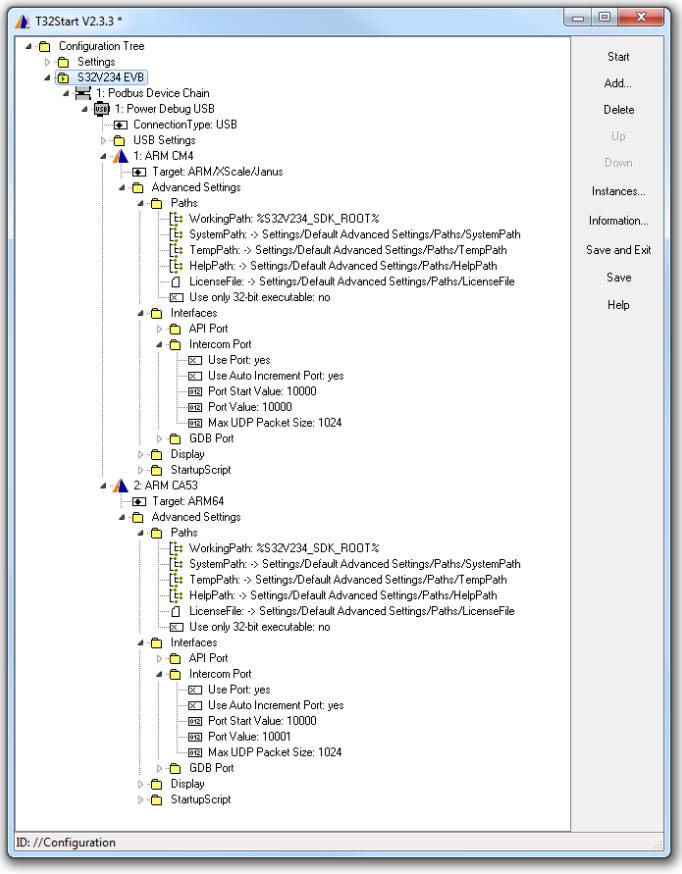

2. By right-clicking the context menu, add a Podbus Device chain and Power Debug USB/Ethernet to your configuration.



3. Add two cores to the Power Debug tree and set the targets as ARM/Scale/Janus and ARM64 (the order is important).




4. Set the WorkingPath directory and Intercom Port according to the following figure. In order to be compatible with startup script, the Port Values needs to be identical to the settings presented.



# 3.3 Running the applications via Linux OS

The VSDK supports the Linux BSP18.

## 3.3.1Prerequisites

As a prerequisite for a Linux boot, the SD Card is needed. For now, the Class 4 SDHC cards are recommended (tested). Please note, some of the microSDHC cards in SDHC adapter and Class 10 were tested with negative results.

## 3.3.2SD Card preparation

### 3.3.2.1Format the card

- Load the card into the reader
- Get the name of sdcard by using `cat /proc/partitions`
    - in this example, the device is called /dev/sdb. Please change the name in all lines below accordingly.

- `sudo fdisk /dev/sdb`

```
d [repeat this until no partition is reported by the 'p' command ]

n [create a new partition]
p [create a primary partition]
1 [the first partition]
<enter> [using the default value will create a partition that starts at
offset 2048]
+255M [size of the actual partition = 255 MB]

n [create a new partition]
p [create a primary partition]
2 [the second partition]
<enter> [using the default value will create a partition that starts at
offset 67584]
<enter> [using the default value will create a partition that uses the
remaining space on the card]

t [set partition type]
1 [partition #1]
c [FAT32]
t [set partition type]
2 [partition #2]
83 [Linux]

w [ this writes the partition table to the medium and fdisk exits]
```

- Remove the SD card from the slot and put it back again

- `sudo mkfs.vfat -n boot /dev/sdb1`
- `sudo mkfs.ext3 -L rootfs /dev/sdb2`

- Remove the SD card from the slot and put it back again. Two file systems should be automatically loaded.

### 3.3.2.2Load the content onto the card

The VSDK contains U-boot, Linux kernel, devicetree and Linux root file system, which were tested with VSDK SW on s32v234-evb board. To use them, please unpack the build_content.tar.gz file – build_content directory will be created.

- `cd <path_to_build_content>/v234_linux_build/s32v234evb/boot`
- `sudo dd if=u-boot.s32 of=/dev/sdb bs=512 seek=8 conv=fsync`
- `sudo cp Image /media/boot`
- `sudo cp s32v234-evb.dtb /media/boot`
- `cd /media/rootfs`
- `sudo tar -xvf <path_to_build_content>/v234_linux_build/s32v234evb/rootfs/rootfs-evb.tar.gz ./`
- `sync`

Linux kernel (Image), devicetree ( s32v234-evb.dtb) and root-file system can be used directly from current Linux BSP code drop.

### 3.3.2.3Copy the user built content onto the card

- User can copy anything to the /media/rootfs - this is the root file system of the board and the files will be visible

### 3.3.2.4Boot the board

- Insert the SD card into the EVB and turn it on. The UART is running on 115200 bps.
- The Linux will boot into command line and automatically loads all necessary drivers.

# 4 Connection to the host PC

## 4.1.1UART

Following chapter describes UART connectivity to the host PC. The UART enables the user to use standard out C functions (printf) for console output.

### 4.1.1.1Windows OS

To be able to connect the board to the host PC running on Windows OS, the USB to UART bridge VCP Driver needs to be installed:

1. Download the driver from http://www.silabs.com/products/mcu/pages/usbtouartbridgevcpdrivers.aspx
2. Turn on the boards
3. Install the Driver
4. Open Device Manager and find the Silicon Labs USB to UART Driver

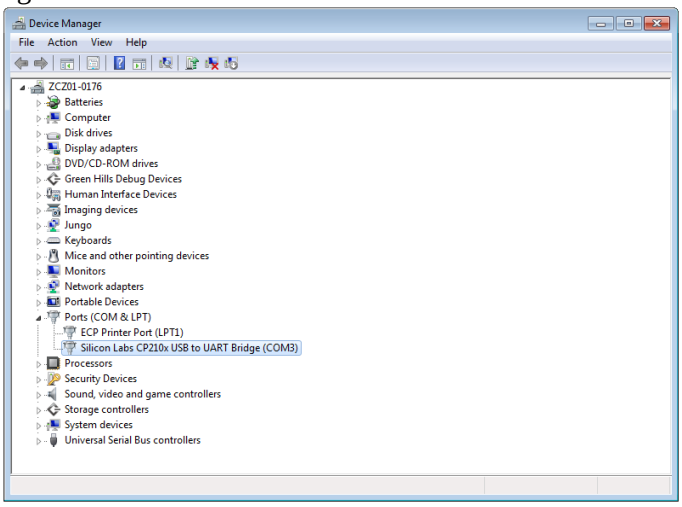



**Figure 10** USB to UART Driver in Device manager

5. In Port Settings in the Driver's properties, set following settings:
   - Bits per second: 115200
   - Data bits: 8
   - Parity: None
   - Stop bits: 1
   - Flow control: None

After successful setup of the driver, it is possible to connect turned-on boards with console client application (e.g. putty) on **`COMX`** and **`115200 bps`** (**`115200 bps`** in case of Linux OS).

#### 4.1.1.2Linux host

The Linux OS has in-built drivers for USB serial connection. After turning up the boards, it's possible to connect the console (e.g. putty) on **`/dev/ttyUSB0`** on **`115200 bps`** (**`115200 bps`** in case of Linux OS).