AN13119

i.MX Power Measurement Tool

Rev. 0 — 04/2021 Application Note

Contents

1 Introduction

The Power Measurement Tool (PMT) provides general control of NXP i.MX8 evaluation kits, collects on-board power consumption information, and provides reports to the user. It integrates a user-friendly GUI with many features, as well as a TUI for command-line operation.

All operations are performed through the USB debug connection on compatible and supported evaluation kits.

The PMT is implemented in Python and actively supported for Linux OS distributions (Ubuntu 16.04 and above) and the Windows OS (Windows 10 and above).

1.1 Acronyms and definitions

ADC	Analog-to-Digital Converter
BCU	Board Control Utilities
EEPROM	Electrically Erasable Programmable Read-Only Memory
FTDI	Future Technology Devices International
GPIO	General-Purpose Input Output
GUI	Graphical User Interface
I^2C	Inter-Integrated Circuit
MPSSE	Multi-Protocol Synchronous Serial Engine
TUI	Text User Interface
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus

1.2 Architecture

Figure 1 shows the power measurement hardware architecture of the NXP i.MX8 evaluation kits family.

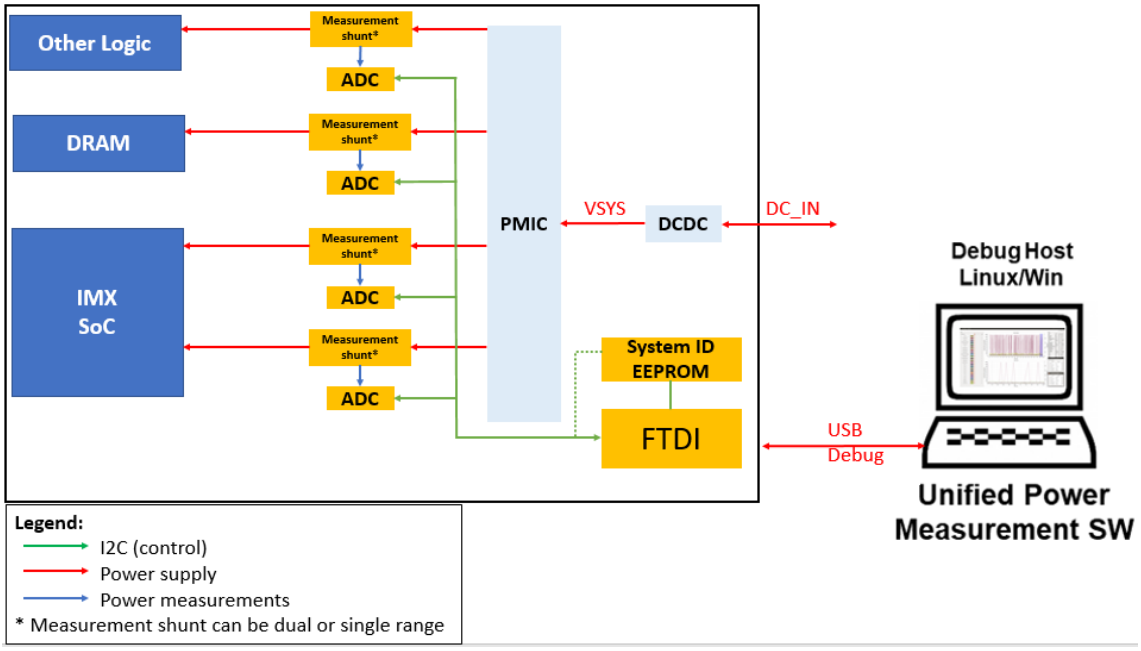


Figure 1. Power measurement hardware architecture

The host PC communicates with the board through the USB debug cable. The USB debug port is connected on-board to a FT4232 USB-to-quad UART transceiver. The FT4232 chip can also emulate other protocols (I^2C in particular, which is used by the PMT to control various board components). The board also embeds a system ID EEPROM (referenced to as EEPROM in the rest of the document) connected either to an I^2C debug interface or to an FTDI chip depending on the board for storing useful board information. The PMT leverages this EEPROM to detect the board type and configuration at startup.

Power measurements are performed by measuring the voltage drop across a high-precision shunt resistor using the on-board ADC. The on-board ADCs used are the four-channel PAC1934 energy-monitoring devices.

For power rails with high minimum/maximum load variations (for example, power consumption in standby mode vs active mode), a dual-range measurement mechanism using high-current and low-current shunt resistors can be implemented to improve the measurement precision. Switching between the high-current and low-current modes is controlled by the user via the PMT application. For the other rails, only one single-range shunt resistor is used (see Figure 2).

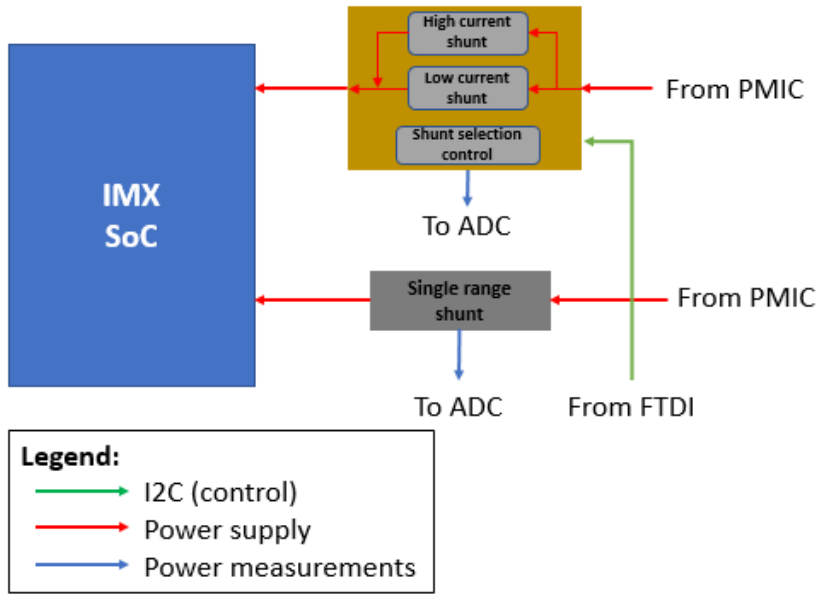


Figure 2. Measurement shunt resistors options

1.3 Supported platforms

The PMT currently supports the following evaluation kits:

Table 1. Supported evaluation kits

Board name	Number of rails	--board ID
iMX8MPLUSLPD4-PWR Rev. A0	27	imx8mpevkpwra0
iMX8MPLUSLPD4-PWR Rev. A1	27	imx8mpevkpwra1
iMX8DXL-EVK	24	imx8dxlevk
iMX8ULP-EVK	15	imx8ulpevk

1.4 License

The PMT is licensed under the BSD-3 Clause New or Revised License.

2 Installation

2.1 Download information

The PMT source code is available on GitHub and it can be cloned or downloaded as a ZIP file.

```
git clone https://github.com/NXPmicro/pmt.git
```

See *README.md* for more information.

2.2 Installation on Linux OS distributions

Python 3 is required for the Linux OS distributions (at least version 3.5.4).

The following packages must be installed to satisfy the dependencies of the tool:

- pylibftdi
- numpy
- pyqtgraph
- Pyqt5
- oyaml

The installation of dependencies can be done as follows:

```
pip3 install -r requirements.txt
sudo apt install python3-pyqt5 libftdi1-dev
```

To access the FTDI chip, run the PMT as a superuser (not recommended) or use the following procedure, adding specific udev rules to allow the FTDI chip to access members of the dialout group (or any other groups you may chose):

1- Create a file named */etc/udev/rules.d/90-ftdi.rules*, containing the following line:

```
SUBSYSTEMS=="usb", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6011", GROUP="dialout",
MODE="0660"
```

2- Then reload the udev rules as follows:

```
$ sudo udevadm control --reload-rules
$ sudo udevadm trigger
```

3- Add your user account to the standard dialout group:

```
$ sudo usermod -aG dialout $(echo $(whoami))
```

4- logout of your machine, and login again.

2.3 Installation on Windows OS host

Install Python version 3.x (at least 3.5.4): www.python.org/downloads/windows/.

During the installation, click “add Python 3.x to PATH” and check the version after the installation.

The following packages must be installed for the tool to start:

- pip3 install windows-curses
- pip3 install ftd2xx
- pip3 install numpy
- pip3 install pyqtgraph
- pip3 install pyqt5
- pip3 install oyaml

Install the FTDI D2XX library from www.ftdichip.com/Drivers/D2XX.htm. Copy the files into the PMT folder.

2.4 EEPROM configuration and usage

2.4.1 Description

The board includes a system EEPROM for storing configuration information. The PMT leverages this EEPROM to detect the board type and configuration at startup.

The EEPROM (I^2C or FTDI) stores the useful information as follows:

- Board ID and revision
- SoC ID and revision
- PMIC ID and revision
- Number of measurable power rails on the board
- Board serial number

When using the board for the first time, the user must ensure that the EEPROM is correctly programmed (EEPROMs are not programmed during board manufacturing). If the EEPROM is not programmed, an error may be raised during PMT startup, as described in the EEPROM Usage section. An example of the evaluation kit configuration information stored in the EEPROM is shown below. This is just an example. Ensure that the parameters are correctly set in the EEPROM_Programmer_Tool yaml file for your board. When the configuration is complete, it can be flashed into the EEPROM (see EEPROM configuration).

To program the EEPROM, use the EEPROM_Programmer_Tool yaml file provided on GitHub (inside the *Docs* folder).

- BOARD_ID: NXP i.MX8MP EVK PWR board
- BOARD_REV: A1
- SOC_ID: i.MX8MP
- SOC_REV: A1
- PMIC_ID: PCA9450CHN
- PMIC_REV: NOT FOUND
- NBR_PWR_RAILS: 27
- BOARD_SN: 1

2.4.2 EEPROM configuration

Flash the board configuration data into the EEPROM:

```
$ python3 main.py eeprom -m write -f docs/EEPROM_Programmer_Tool.yaml
```

```
** Info collected.

/!\ You are going to overwrite EEPROM content, want to continue? Y/y/N/n
Y
I2C EEPROM - Which board are you using ? ( (1) imx8dxlevk / (2) imx8mpevkpwr(a0 / a1) )
2
** Writing to I2C EEPROM ...
** Done.
```

Figure 3. EEPROM content flashing example

The EEPROM content can be read using the following instruction:

```
$ python3 main.py eeprom -m read
```

```
I2C EEPROM - Which board are you using ? ( (1) imx8dxlevk / (2) imx8mpevkpwr(a0 / a1) )
2
** Reading I2C EEPROM ...

CONFIG_FLAG: Programmed EEPROM
BOARD_ID: NXP i.MX8MP EVK PWR Board
BOARD_REV: A1
SOC_ID: i.MX8MP
SOC_REV: A1
PMIC_ID: PCA9450CHN
PMIC_REV: Unknown
NBR_PWR_RAILS: 27
BOARD_SN: 1

** Done.
```

Figure 4. EEPROM content reading example

Note: The EEPROM-flashing requirement and content organization is common across all supported boards and power measurement applications, including BCU.

2.4.3 EEPROM usage

Depending on the number of evaluation kits connected to the host PC, complementary information may be passed through the command line (board ID or/and location ID).

Assuming that a single evaluation kit is connected:

- Specify the board ID in the command line:
  - The PMT checks the EEPROM content of the connected board:
    - If the board ID in the EEPROM matches the board name from the command line, the PMT tool is started using the corresponding board's configuration.
    - If the board ID in the EEPROM is blank, the PMT starts using the board's configuration from the command line and a warning is raised to program the EEPROM.
    - If the EEPROM is incorrectly flashed and the board ID in the EEPROM does not match the board ID from the command line, a warning is raised and the PMT starts using the board configuration specified in the command line.
- No board ID is specified in the command line and the EEPROM:
  - is correctly programmed:
    - The PMT starts using the EEPROM board's configuration.
  - is not programmed:
    - The PMT raises an error and the EEPROM must be programmed or the board ID must be specified in the command line.

If two or more boards are connected to the same host PC, the PMT needs additional info (board ID or/and location ID) to determine which board to connect to. The boards' EEPROMs are considered to be configured as required.

- When only one board of the desired board ID is connected to the debug host, specify the board ID or the location ID in the command line.

When two or more connected boards are of the same type, specify the desired board's location ID in the command line. Boards location ID can be obtained by the lsftdi command, described below.

3 Program usage

3.1 Program commands

Commands	Description
lsftdi	Shows the list of connected boards and their location IDs.
lsgpio [-b] [-i]	Shows the list of the available GPIO pin of the board specified by the board ID or the location ID.
lsboard	Lists all the supported board IDs.
lsbootmode [-b] [-i]	Shows the list of the available boot mode of the board specified by the board ID or the location ID.
reset [-b] [-i] [-bootm] [-d]	Resets the specified board with the possible boot mode and delay selected.
set_gpio [-b] [-i] [-g] [-v]	Modifies the GPIO with the specified value.
monitor [-b] [-i] [-m] [-l] [-t] [-d]	Displays the power information in the GUI or TUI.
eeprom [-m] [-i] [-f]	Reads or writes into the FTDI/I^2C EEPROM.
server [-i] [-b] [-p]	Use the PMT as a server and send the power data through the network with a specified port.

3.2 Program options

- General options:

Options	Description
`-b / --board`	Sets the board ID.
`-i /--id`	Sets the location ID.

- reset command:

Options	Description
`-bootm / --boot_mode`	Sets the boot mode name for the booting method.
`-d / --delay`	Specifies the delay (in seconds) before resetting the board.

- set_gpio command:

Options	Description
`-v / --value`	The desired value for the GPIO (1/0/low/high/toggle).

Table continues on the next page...

Table continued from the previous page...

Options	Description
`-g / --gpio_name`	The name of the GPIO.

- monitor command:

Options	Description
`-m / --mode`	The monitor mode, GUI or TUI. TUI is the default option.
`-l / --load`	The CSV/PMT file to load (only in GUI mode).
`-t / --time`	The monitor during the specified time (only in TUI mode).
`-d / --dump`	Saves the data into a file (CSV is the default extension; only in TUI mode).

- eeprom command:

Options	Description
`-m / --mode`	The EEPROM mode; read or write.
`-f / --file`	Specifies the path to the file to write into the EEPROM.

- server command:

Options	Description
`-p / --port`	The port to open.

3.3 Usage examples

The PMT can automatically detect the connected board(s) by reading the EEPROM content, as described in the EEPROM configuration and usage section. If two (or more) boards are connected to the same debug host, specify at least the board ID or the location ID (provided by the “lsftdi” command) of the given board.

NOTE

For the Windows OS, use the “python” command instead of the “python3” command.

For the following examples, it is assumed that two boards are connected. Specify at least one board ID or location ID.

List the connected board:

```
$ python3 main.py lsftdi
```

List the available GPIOs for the iMX8DXLEVK board:

```
$ python3 main.py lsgpio -b imx8dxlevk
```

List the boards supported by the PMT:

```
$ python3 main.py lsboard
```

List the boot mode for the iMX8DXLEVK board:

```
$ python3 main.py lsbootmode -b imx8dxlevk
```

Reset the iMX8DXLEVK board after three seconds and boot from EMMC:

```
$ python3 main.py reset -b imx8dxlevk -bootm emmc -d 3
```

Set the FT_GPIO1 (name listed via the “lsgpio” command) to a high level on the iMX8DXLEVK board:

```
$ python3 main.py set_gpio -b imx8dxlevk -g FT_GPIO1 -v 1
```

Monitor in the GUI:

```
$ python3 main.py monitor -m gui
```

Monitor in the TUI for 35 seconds and dump the data to the *test.csv* file:

```
$ python3 main.py monitor -t 35 -d test.csv
```

Read the EEPROM content:

```
$ python3 main.py eeprom -m read
```

Flash the EEPROM content:

```
$ python3 main.py eeprom -m write -f docs/EEPROM_Programmer_Tool.yaml
```

Use the PMT as a server with port 65432:

```
$ python3 main.py server -b imx8dxlevk -p 65432
```

4 PMT monitor modes

The PMT supports two monitor modes (GUI and TUI). The PMT configuration is identical in the GUI and TUI modes.

4.1 PMT configuration

4.1.1 Power rails selection

The power rails to monitor are configured via the “RAILS_TO_PROBE” list in the *program_config.py* file. By default, the program probes every power rail available on the board except for the power groups, with “RAILS_TO_PROBE” set to “all”. If you want to probe only a subset of available rails or power groups (sum of the specific power rails for a system, e.g. “GROUP_PLATFORM”), update “RAILS_TO_PROBE” with the desired rails’ names, for example:

```
RAILS_TO_PROBE = ['5V0', '3V3_USB', 'VDD_SNVS1', 'GROUP_PLATFORM']
```

The PMT always samples data at the highest rate possible. Reducing the number of rails/groups to probe globally improves the sampling rate.

4.1.2 Shunt resistor configuration

It is possible for some rails to switch between a high-current shunt and a low-current shunt (and the other way round) to improve the accuracy of the measured value. The rails supporting this feature can be found in the board configuration file of each board. For example, with iMX8MPPWR-EVK, the following is in the “mapping_power” list of the *imx8mpevkpwra1.py* file in the *board_configuration* folder:

```
{'name': 'VDD_PLL_ANA_1V8', 'ftdi': [1, 0x60, 0x40], 'pca9548': [1, 0x71], 'pac': [4, 0x15, 5],
'rsense': [1008, 11000, 1], 'rsense_ctl': 'FT_IO_05'}
```

This means that the rail can switch between 1008 mOhm in the high-current shunt and 11000 mOhm in the low-current shunt, thanks to “GPIO "FT_IO_05”, which controls the switch load.

Figure 5 shows the corresponding schematic of the switch load for the “VDD_PLL_ANA_1V8” rail. R94 and (R91+R94) are respectively the high and low current shunt resistors values.

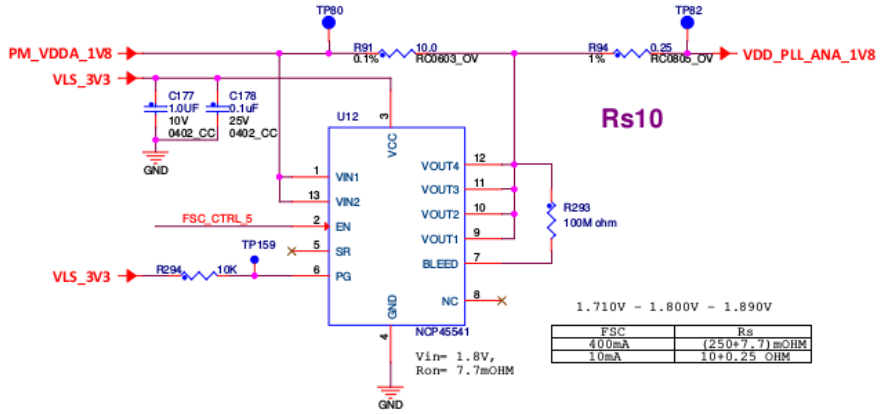


Figure 5. Range switching circuit on i.MX8MPEVK-PWR

NOTE

If the shunt value was physically changed on the evaluation kit, update the board configuration file with the new and correct shunt value.

To ensure that the sense voltage across the shunt does not exceed the ADC input maximum range, the switch from high-current shunt to the low-current shunt may be prohibited by the PMT.

To switch from the high-current shunt to the low-current shunt (via TUI or GUI), the program compares the average of the last 300 ms of data current with the computed limit current threshold for the rail. The current limit threshold is computed by the PMT using the rail’s maximum current limit with the low-current shunt (100 mV/low-current shunt value) plus the margin defined by the protection offset in the *program_config.py* file (“LOW_SWITCH_RESISTANCE_OFFSET”). If the average is under the computed limit, the switch is authorized and performed by the PMT. Otherwise, it is prohibited.

NOTE

On the opposite, the PMT does not do any pre-check to switch from the low-current shunt to the high-current shunt. It is always permitted, because it is always safe for the ADC.

Example:

If the current limit is 10 mA and the offset is 10 %, the average measured value of the current should be lower than 9 mA to allow the switching from the high-current shunt to the low-current shunt. This offset can be modified according to your needs.

4.1.2.1 GUI mode

The GUI interface provides direct user control over the power rails with the dual measurement range. To switch from the high-current shunt to the low-current shunt in the GUI mode (and the other way round), click on the rail name of interest and then click “Switch resistance” (only available on the rail supporting the high-/low-current shunt feature on the board).

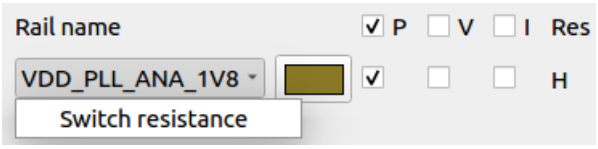


Figure 6. “Switch resistance” menu

If authorized, the “Res” field of the rail changes from:

- “H” (high-current shunt):

Figure 7. High-current shunt used on VDD_PLL_ANA_1V8 rail (example)

- to “L” (low-current shunt):

Figure 8. Low-current shunt used on VDD_PLL_ANA_1V8 rail (example)

4.1.2.2 TUI mode

The TUI interface also provides direct user control over the power rails with dual measurement range. Enter the power rail location value to switch the sense resistor to the desired value.

Example:

Press “A” to switch the current range for “VDD_PLL_ANA_1V8” from the high-current shunt (1008 mOhm) to the low-current shunt (11000 mOhm).

If authorized, the resistance value (in blue) of the rail switches from:

- the high-current shunt value (lower resistor value):

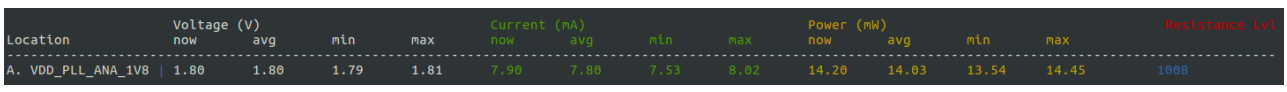


Figure 9. High-current shunt used on VDD_PLL_ANA_1V8 rail (example)

- to the low-current shunt value (higher resistor value):

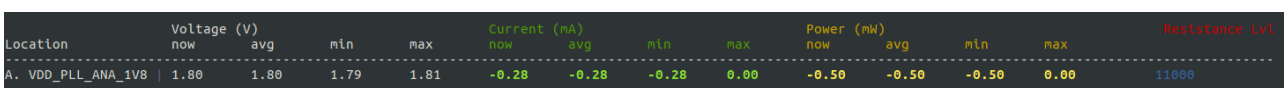


Figure 10. Low-current shunt used on VDD_PLL_ANA_1V8 rail (example)

Press “A” again to switch back to 258 mOhm. There is no check in the PMT to switch from the low-current shunt to the high-current shunt and it is always authorized.

4.2 GUI mode

Figure 11 is a screenshot of the typical PMT application, showing the main always-on control window and the optional power data windows.

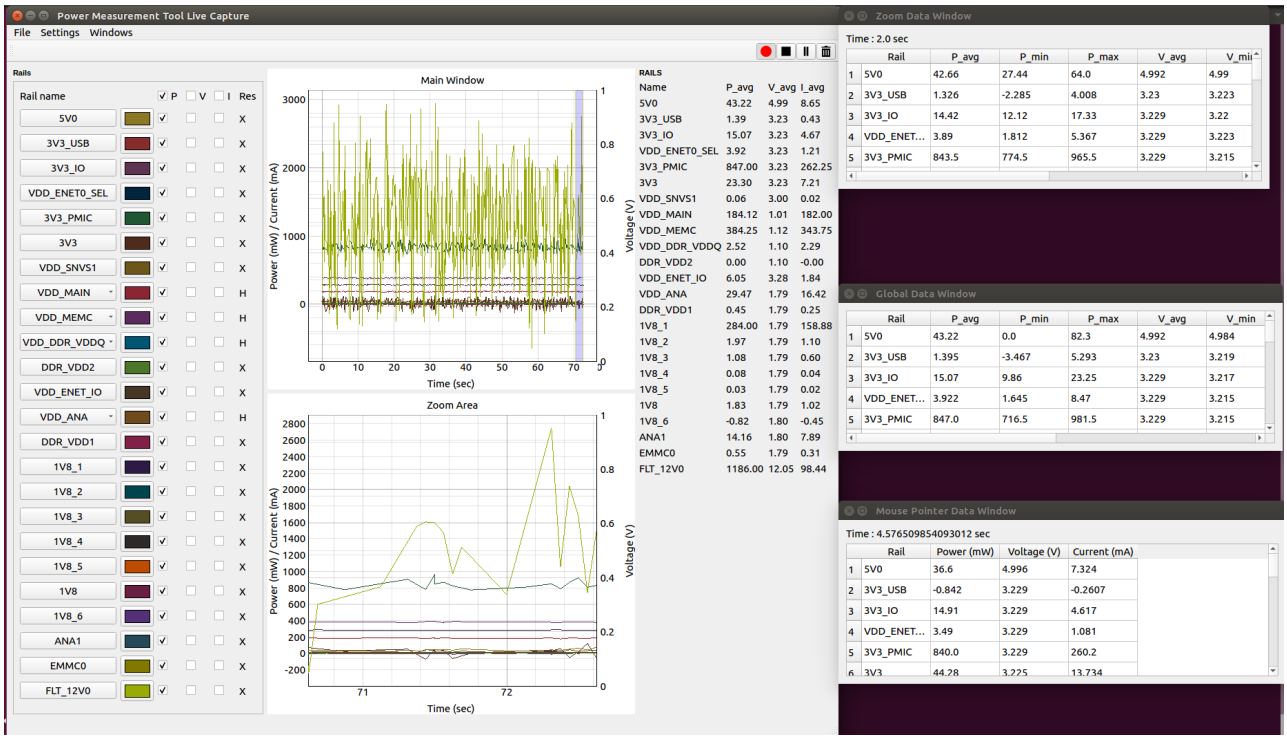


Figure 11. Full PMT GUI view example (iMX8DXL-EVK board)

The supported features are as follows:

- Offline monitor without any board connected.
- Possibility to import a CSV file from the BCU tool.
- PAC's bipolar mode. The PAC can sense from -100 mV to 100 mV represented with a 16-bit two’s complement instead of from 0 mV to 100 mV.
- PAC's hardware filter. The voltage and current values read from the “avg” register contain a rolling average of the eight most recent VBUS/VSENSE results.
- Stopping/resuming the data acquisition with the stop region displayed.
- Exporting files as CSV, binary PMT, and PNG.
- Importing the CSV and PMT files.
- Selecting the power/voltage/current rails to plot.
- Changing the color of each rail.
- Resetting the capture.
- Pausing the display refresh.
- Hiding all rails plotted with one click.

- Additional window with the mouse pointer data information.
- Additional window with the zoom region minimum/maximum/mean values of the power/voltage/current.
- Additional window with the minimum/maximum/mean values of the power/voltage/current since the start of the application.
- Hiding/showing external windows.
- Switch resistor value of each rail (when available).

4.3 TUI mode

Figure 12 is a typical screenshot of the TUI.

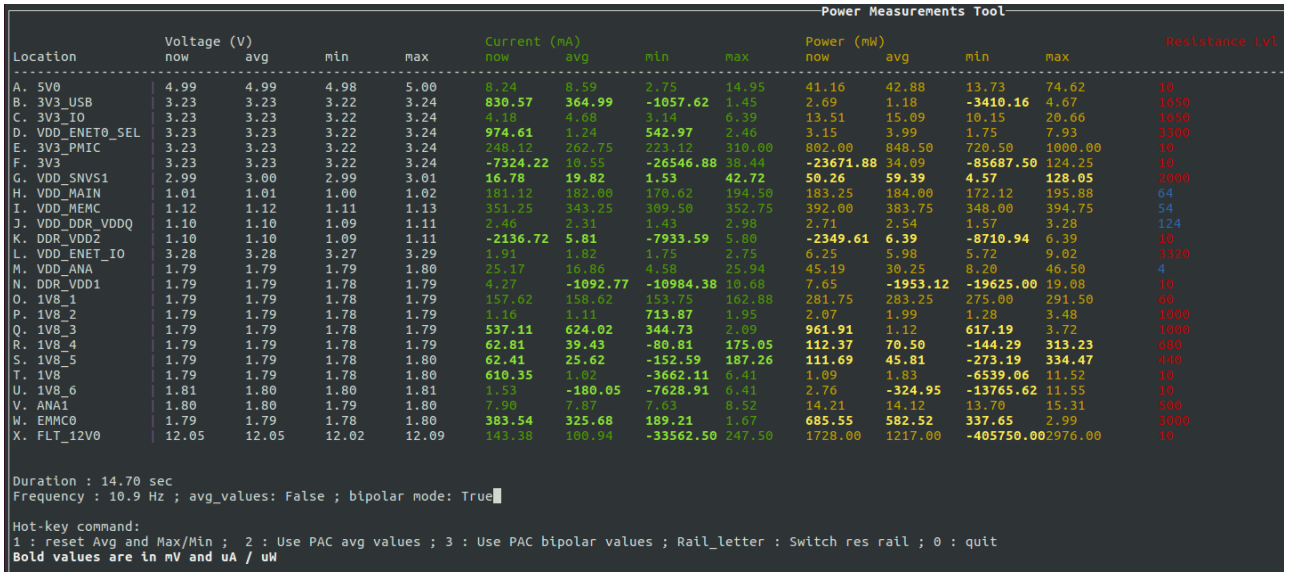


Figure 12. TUI view

The default values are in V, mA, and mW. If the values are lower than 1 V/1 mA/1 mW, the program switches to a new range (mV, uA, uW) and displays these values in bold.

The supported features are as follows:

- PAC's bipolar mode. The PAC can sense from -100 mV to 100 mV, represented with a 16-bit two's complement instead of from 0 mV to 100 mV.
- PAC's hardware filter. The voltage and current values read from the "avg" register contain a rolling average of the eight most recent VBUS/VSENSE results.
- Displaying the minimum/maximum/mean values of the voltage/current/power.
- Exporting to a file (CSV format if no extension is specified).
- Switching the resistor value of each rail when available (the resistor value is in blue).
- Resetting the capture.
- Displaying the frequency of the captured data.
- Displaying the duration since the application starts.

The supported hot keys are as follows:

- Press 1 to reset the capture.
- Press 2 to use the PAC's average values.
- Press 3 to switch between the PAC unipolar/bipolar modes.

- Press the “Location” letter to switch from/to the high-current shunt to/from the low-current shunt resistors of the corresponding rail.
- Press “0” to quit the application.

NOTE

The PAC's bipolar mode (enabled by default) reports negative values. This features allows for higher precision in low-power modes, because it can sense the signal noise or capacitor charge/discharge (for example). However, you can easily switch to the unipolar mode in different UIs.

5 PMT server mode

The PMT has a server feature that allows the tool to collect power data and send them through the network. Implement your own client program to communicate with the PMT using the TCP protocol. The port to open must be the same between the server and client. Each time the client side makes a request to the server (i.e non-empty string), the server sends all the power data in a pre-defined standard format. The format is `rail_name1:power1;rail_name2:power2`.

The data-collection thread runs in background, waiting for a request from a client. The sent power data are an average depending on the periodicity of the client's request.

NOTE

The PMT server can accept a maximum of 10 clients simultaneously. At the end of the connection, close the server program by pressing Ctrl+C. If the client stops the communication, the server is still running (useful in cases with multiple clients).

For example, a simple client program is created. It opens a connection, specifying the correct IP address of the server and the port to open. Then it makes a request and displays the received content every second. Thus, the power data for each rail are an average during one second.

On the server side, start the PMT tool in the server mode with the following command:

```
python3 main.py server -p 65432
```

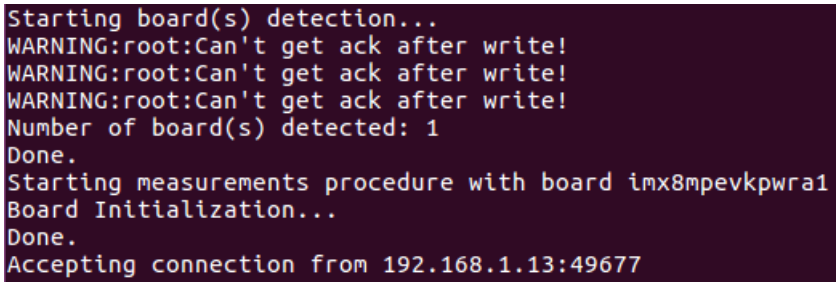


Figure 13. PMT tool start

After starting the client program, you can see an example of the content received every second, containing the power data of each rail.

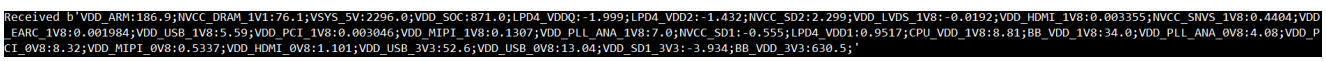


Figure 14. Client side

Stop the communication by pressing Ctrl+C on the server side.

6 Usage example: running PMT on iMX8DXL-EVK board

Follow the below instructions to perform a measurement in GUI mode on the iMX8DXL-EVK board.

6.1 Connection setup

Connect the micro-B USB debug port (J19) to the host PC and the 12-V DC IN power supply to J1. Open a console terminal to the platform (115200 8, N, 1, no flow control).

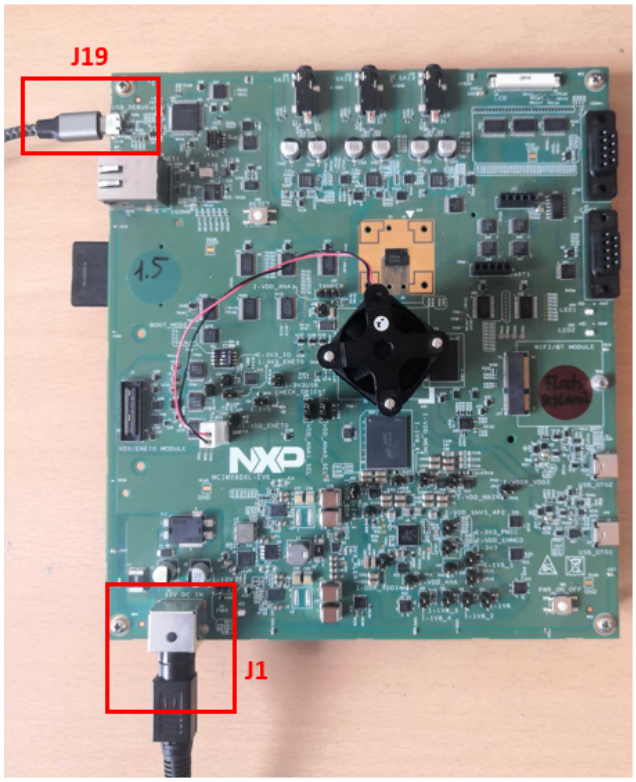


Figure 15. iMX8DXL-EVK board

6.2 EEPROM check

Check if the board's EEPROM is empty or not as follows:

```
$ python3 main.py eeprom -m read
```

```
I2C EEPROM - Which board are you using ? ( (1) imx8dxlevk / (2) imx8mpevkpwr(a0 / a1) )
1
** Reading I2C EEPROM ...

CONFIG_FLAG: Programmed EEPROM
BOARD_ID: Unknown
BOARD_REV: Unknown
SOC_ID: Unknown
SOC_REV: Unknown
PMIC_ID: Unknown
PMIC_REV: Unknown
NBR_PWR_RAILS: Unknown
BOARD_SN: Unknown

** Done.
```

Figure 16. Check EEPROM content

As shown in Figure 16, the EEPROM is empty. Set the expected board parameters in the docs/EEPROM_Programmer_Tool.yaml file and flash the EEPROM (see the example in Figure 17).

```
#     - NXP i.MX8MP EVK Board
#     - NXP i.MX8MP EVK PWR Board
#     - NXP i.MX8MP DDR3L Board
#     - NXP i.MX8MP DDR4 Board
#     - NXP i.MX8ULP EVK Board

BOARD_ID: NXP i.MX8DXL EVK Board

# Complete BOARD_REV with revision of the board:
#     - A0 to A4
#     - B0 to B4
#     - C0 to C4
#     - "NOT FOUND" if no information

BOARD_REV: B0

# Complete SOC_ID with rwith following options:
#     - i.MX8DXL
#     - i.MX8MP
#     - i.MX8ULP

SOC_ID: i.MX8DXL

# Complete SOC_REV with revision of the board:
#     - A0 to A4
#     - B0 to B4
#     - C0 to C4
#     - "NOT FOUND" if no information

SOC_REV: A1

# Complete SOC_ID with rwith following options:
#     - PP7100BVM1ES
#     - PCA9450CHN
#     - PPF7100BMMA2ES
#     - PIMX8UD7DVP10SA

PMIC_ID: PP7100BVM1ES

# Complete PMIC_REV with revision of the board:
#     - A0 to A4
#     - B0 to B4
#     - C0 to C4
#     - "NOT FOUND" if no information

PMIC_REV: B1

# Complete NBR_PWR_RAILS with the number of rails supported by board:
#     - 0 to 30

NBR_PWR_RAILS: 24

# Complete BOARD_SN with the number of rails supported by board:
#     - 0 to 30

BOARD_SN: 4
```

Figure 17. Setting iMX8DXL-EVK board parameters to flash the EEPROM (example)

```
$ python3 main.py eeprom -m write -f docs/EEPROM_Programmer_Tool.yaml
```

```
** Info collected.

/!\ You are going to overwrite EEPROM content, want to continue? Y/y/N/n
y
I2C EEPROM - Which board are you using ? ( (1) imx8dxlevk / (2) imx8mpevkpwr(a0 / a1) )
1
** Writing to I2C EEPROM ...
** Done.
```

Figure 18. Flash board configuration EEPROM

Read the EEPROM content again to confirm that it is flashed as expected.

```
$ python3 main.py eeprom -m read
```

```
I2C EEPROM - Which board are you using ? ( (1) imx8dxlevk / (2) imx8mpevkpwr(a0 / a1) )
1
** Reading I2C EEPROM ...

CONFIG_FLAG: Programmed EEPROM
BOARD_ID: NXP i.MX8DXL EVK Board
BOARD_REV: B0
SOC_ID: i.MX8DXL
SOC_REV: A1
PMIC_ID: PP7100BVM1ES
PMIC_REV: B1
NBR_PWR_RAILS: 24
BOARD_SN: 4

** Done.
```

Figure 19. Check EEPROM content after flashing

6.3 Running PMT in GUI mode

As described in section Power rails selection, you can define the list of rails to probe in the “RAILS_TO_PROBE” list in the *program_config.py* file. By default, all measurable power rails are selected:

```
RAILS_TO_PROBE = ['all']
```

Start the PMT in GUI mode. In this example, the EEPROM content is valid and there is only one iMX8DXL-EVK board connected to the host PC. As described in section EEPROM usage, the board is therefore automatically detected by the PMT tool and the GUI is opened:

```
$ python3 main.py monitor -m gui
```

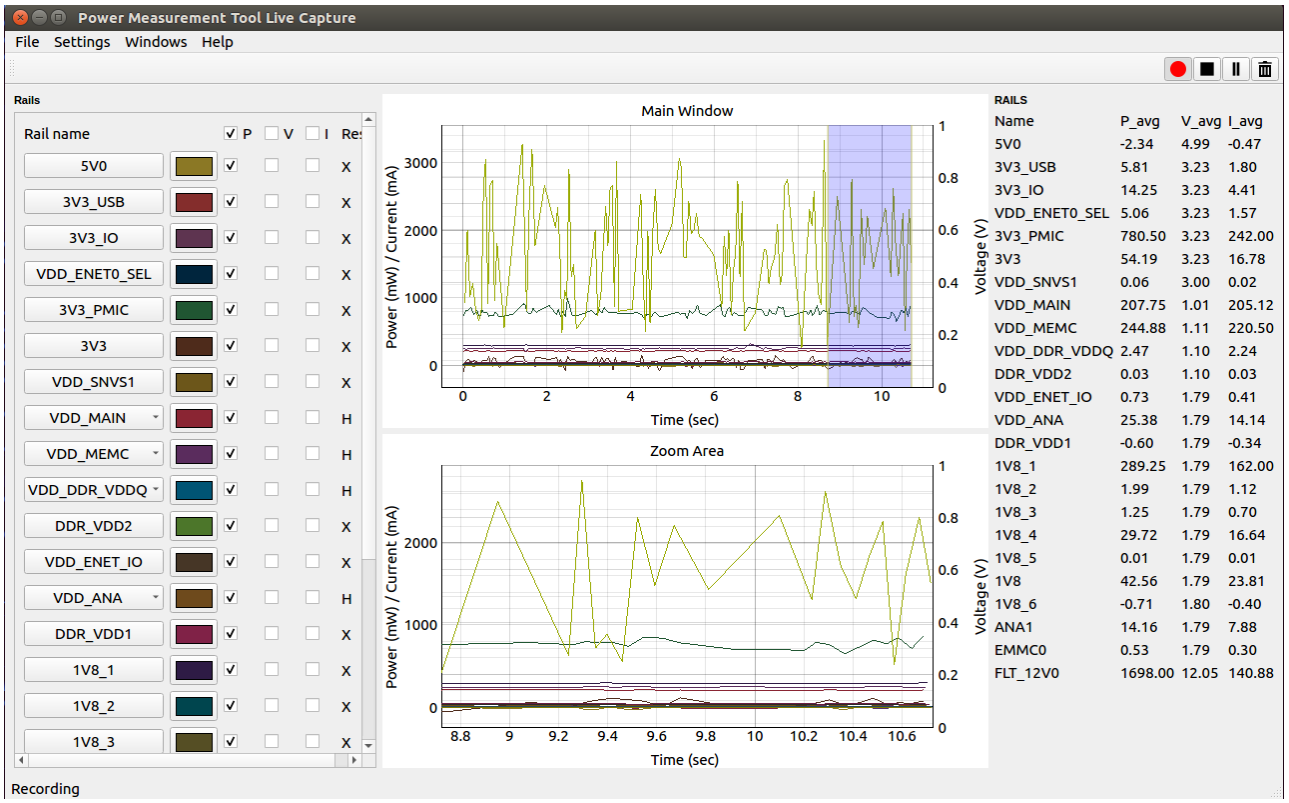


Figure 20. iMX8DXL-EVK GUI view with all rails selected

Quit the application by clicking the close (“X”) button or press Ctrl+C in the terminal.

6.4 Monitor subset of rails in GUI

Reduce the list of rails to probe to the few rails of interest (example only).

The reduced rails of interest are VDD_MAIN, VDD_MEMC, and 3V3_USB and the power groups are “GROUP_SOC” and “GROUP_DRAM”.

Update the “RAILS_TO_PROBE” list in the *program_config.py* file as shown below and save the file:

```
RAILS_TO_PROBE = ['VDD_MAIN', 'VDD_MEMC', '3V3_USB', 'GROUP_SOC', 'GROUP_DRAM']
```

Start the PMT in the GUI mode as follows:

```
$ python3 main.py monitor -m gui
```

In the PMT GUI, only the selected rails are now probed and displayed. All other non-listed rails are not monitored.

Figure 21. PMT GUI with limited rails selection

Open the “Zoom data” window: Click the “Windows” tab in the menu bar and then click “Show / Hide Zoom data window”.

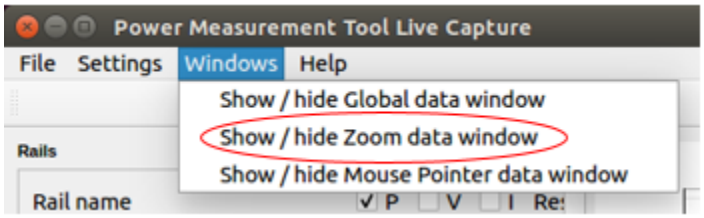
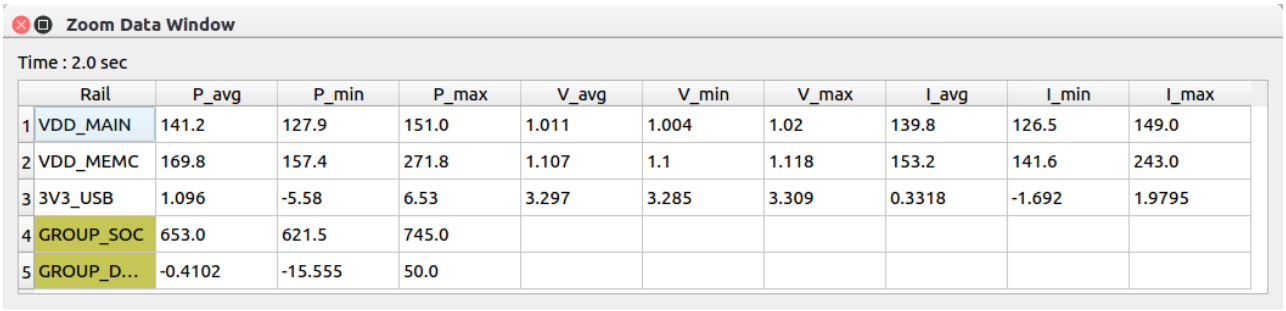


Figure 22. Selecting zoom data windows

The “Zoom Data Window” opens up, giving the measurements of the zoom area (the blue area in Figure 21) selected in the main “PMT GUI Main” window.



	Rail	P_avg	P_min	P_max	V_avg	V_min	V_max	I_avg	I_min	I_max
1	VDD_MAIN	141.2	127.9	151.0	1.011	1.004	1.02	139.8	126.5	149.0
2	VDD_MEMC	169.8	157.4	271.8	1.107	1.1	1.118	153.2	141.6	243.0
3	3V3_USB	1.096	-5.58	6.53	3.297	3.285	3.309	0.3318	-1.692	1.9795
4	GROUP_SOC	653.0	621.5	745.0						
5	GROUP_D...	-0.4102	-15.555	50.0						

Figure 23. View of the external "Zoom Data Window"

In the platform's Linux OS console, enter the following command to send the platform into suspend mode (suspend-to-ram):

```
root@imx8dxlevk:~# echo mem > /sys/power/state
```

The platform enters into suspend-to-ram mode.

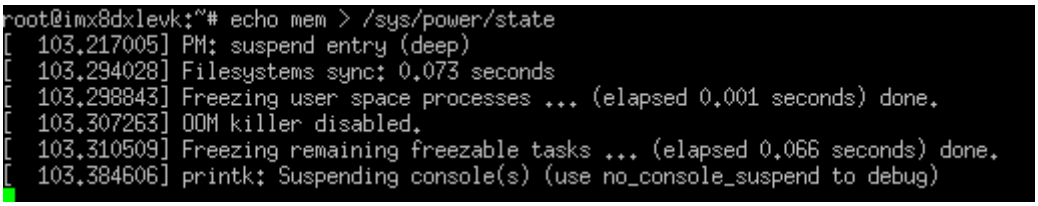


Figure 24. Entry into suspend-to-ram power state

As shown in Figure 24, the power consumption of the monitored rails drops down when the platform enters suspend-to-ram.

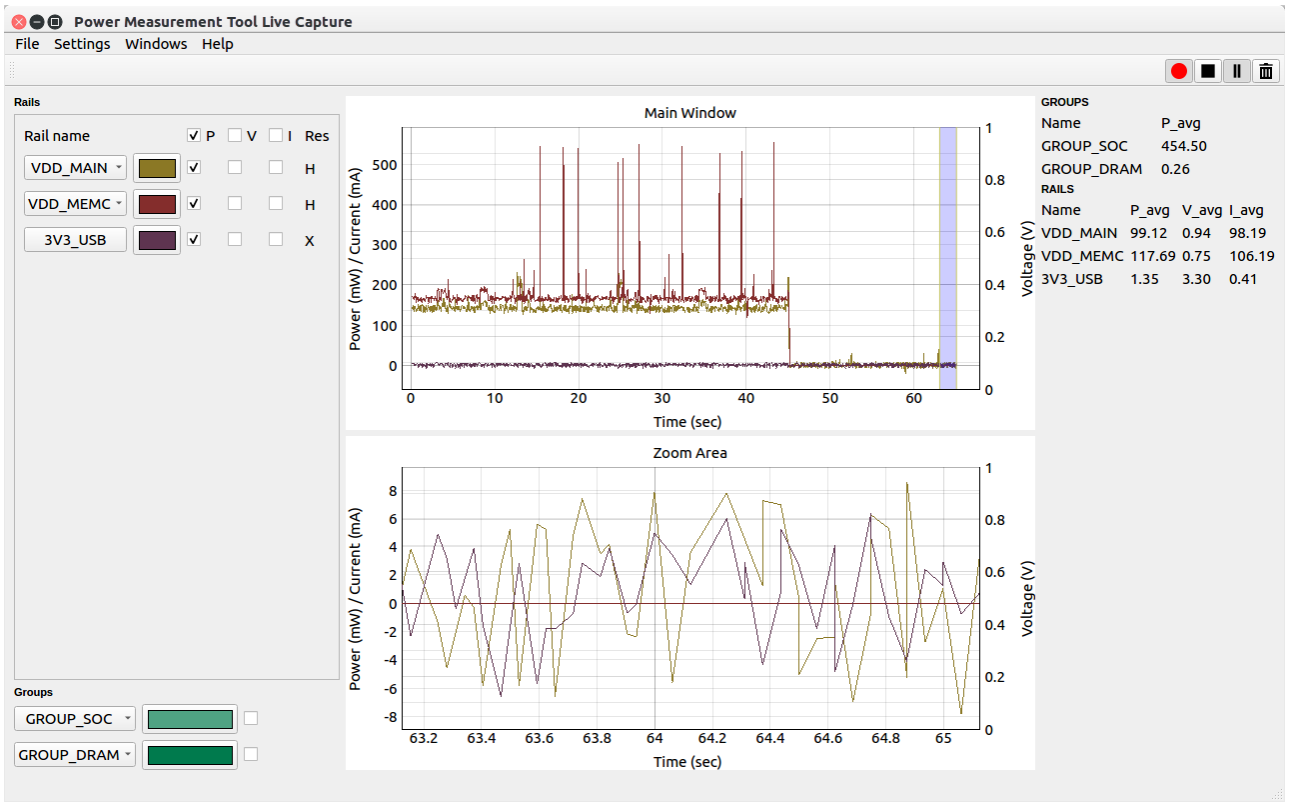


Figure 25. GUI view with evaluation board in suspend mode

The values in suspend mode are shown in the "Zoom Data Window" (Figure 25).

Zoom Data Window

Time : 2.0 sec

	Rail	P_avg	P_min	P_max	V_avg	V_min	V_max	I_avg	I_min	I_max
1	VDD_MAIN	0.9375	-7.836	8.57	0.801	0.794	0.81	1.17	-9.87	10.68
2	VDD_MEMC	-5.32e-05	-0.002483	0.002209	0.001256	0.0	0.006348	-0.005394	-0.8477	0.961
3	3V3_USB	0.7017	-6.594	6.38	3.297	3.287	3.31	0.2126	-2.004	1.938
4	GROUP_SOC	14.805	-10.02	38.28						
5	GROUP_D...	3.129	-16.7	21.34						

Figure 26. Zoom Data Window

To switch the “VDD_MEMC” measurement shunt to the low-current shunt, click on “VDD_MEMC” rail name and then on “switch resistance”.

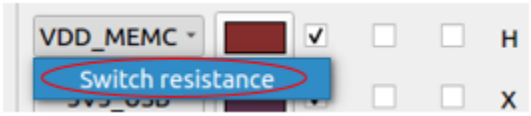


Figure 27. Switch resistance rail's option

The value of the resistance in the GUI switches from “H” to “L” to indicate that the low-current shunt is now used for power measurement on that rail.

Figure 28. Low-current shunt resistor selected

Figure 29 shows the value of the VDD_MEMC rail with the low-current shunt selected and all other rails not displayed in the main GUI, all unselected by unticking their respective “P” power display selection boxes.

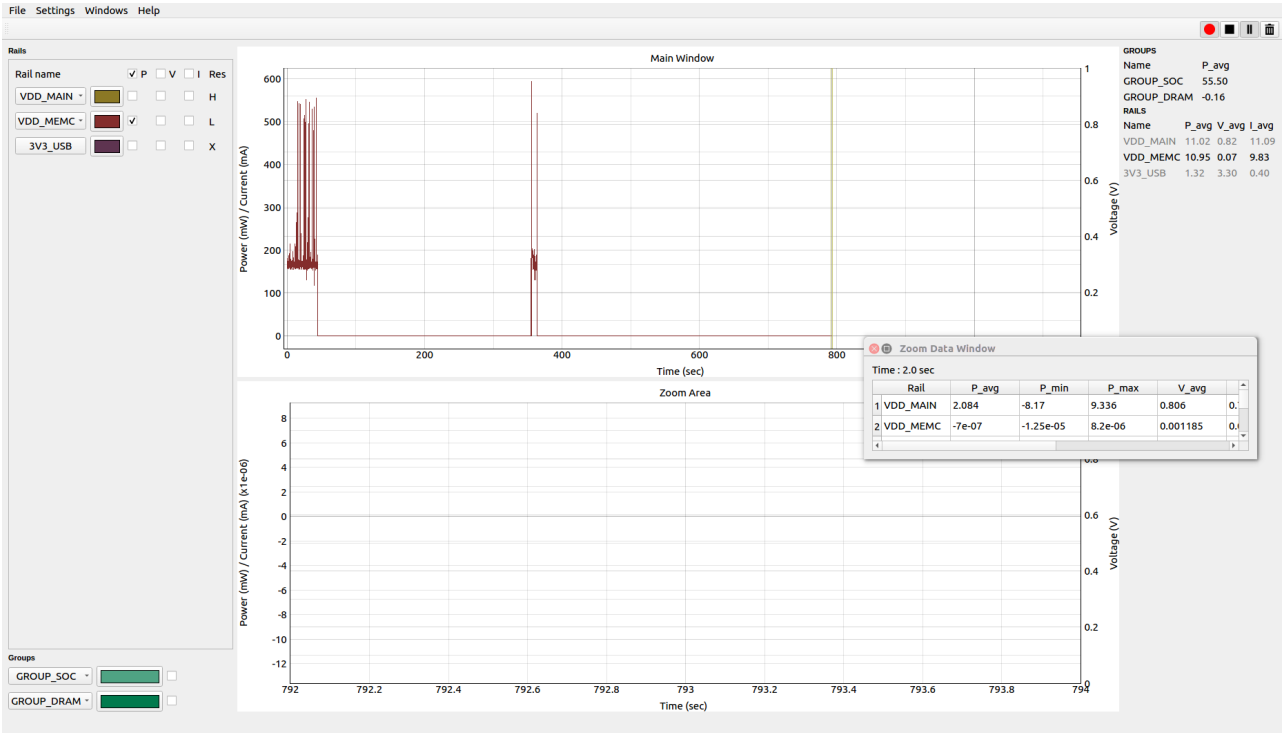


Figure 29. GUI view after switching VDD_MEMC rail resistance

Before exiting from suspend mode, switch back the shunt resistor to the high-current shunt by clicking “Switch resistance” again to avoid a potential crash of the platform due to excessive voltage drop across the shunt resistor and also the saturation of the PAC.

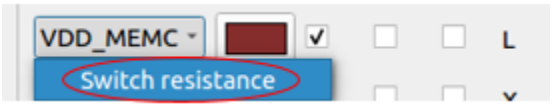


Figure 30. Switch rail resistance back to high-current shunt

Exit the suspend mode by pressing the ON/OFF button (SW3) on the board. The platform exits the suspend-to-ram power state.

```
[  103.384606] printk: Suspending console(s) (use no_console_suspend to debug)
[  103.480621] PM: suspend devices took 0.092 seconds
[  103.487483] Disabling non-boot CPUs ...
[  103.488243] CPU1: shutdown
[  103.489274] psci: CPU1 killed (polled 4 ms)
[  103.490650] Enabling non-boot CPUs ...
[  103.491022] Detected VIPT I-cache on CPU1
[  103.491055] GICv3: CPU1: found redistributor 1 region 0:0x0000000051b20000
[  103.491101] CPU1: Booted secondary processor 0x0000000001 [0x410fd042]
[  103.491898] CPU1 is up
[  103.553650] PM: resume devices took 0.020 seconds
[  103.597606] OOM killer enabled.
[  103.600758] Restarting tasks ... done.
[  103.618328] PM: suspend exit
root@imx8dxlevk:~#
```

Figure 31. Exiting suspend-to-ram power state

The resume of the platform and activity can be observed in the GUI (Figure 32).

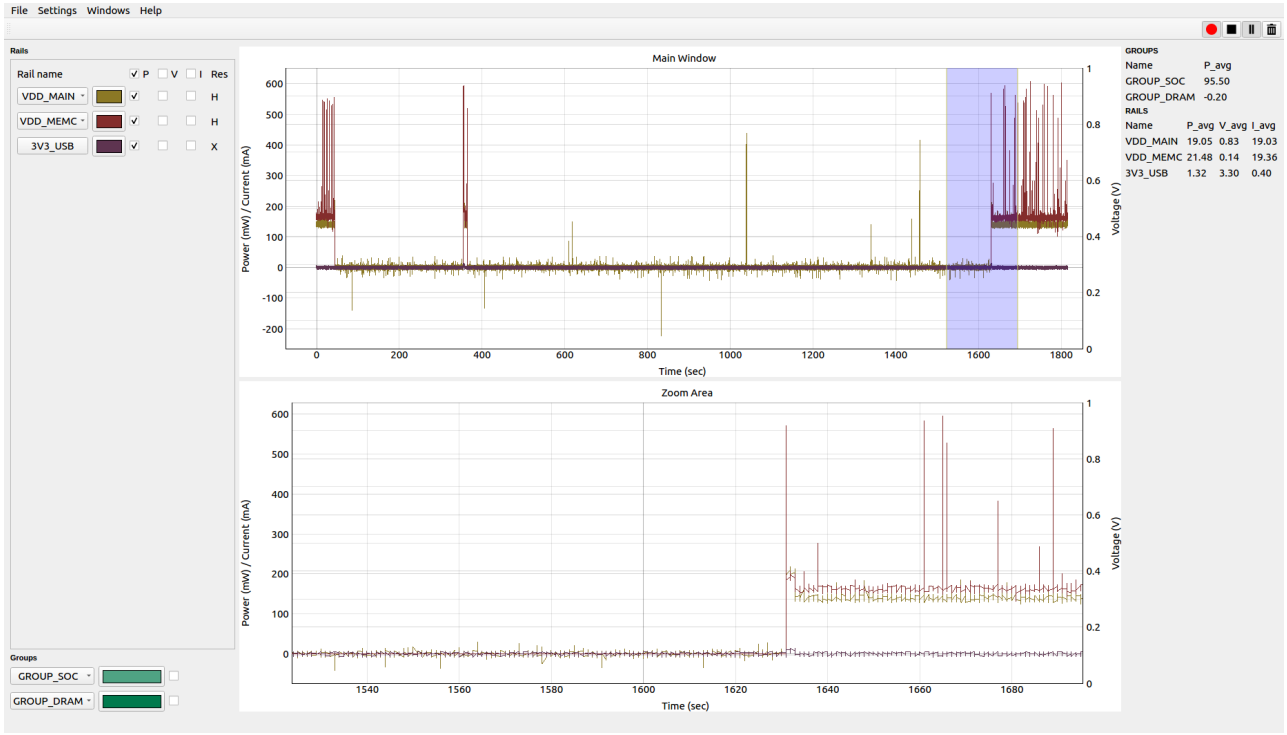


Figure 32. Resume from suspend power state

7 Known issues

iMX8DXL-EVK:

- Rarely after a reset, the board goes into an unstable state and the communication with the GPIO expander PCA6416 (U101) becomes impossible. This part controls the reset GPIOs and switches the resistor GPIOs. In this unstable state, it is not possible to switch between the high-current shunt and the low-current shunt on any rail and reset the board properly.

Workaround:

- Turn off the board and disconnect and reconnect the USB debug cable.

8 Contact

For any questions or issues, please open an issue in PMT Github (https://github.com/NXPmicro/pmt/issues).

9 Revision history

Table 2. Revision history

Revision number	Date	Substantive changes
0	04/2021	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX,EMBRACE, GREENCHIP, HITAG, ICODE, JCOP, LIFE, VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, CodeWarrior, ColdFire, ColdFire+, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, Tower, TurboLink, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© NXP B.V. 2021. **All rights reserved.**

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 04/2021

Document identifier: AN13119

arm