EyePoint IVM API

1.0.2

Generated by Doxygen 1.8.13

Contents

1 File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

2 File Documentation

2.1 ivm.h File Reference

ivm API

```
#include <stdint.h>
#include <wchar.h>
```

Data Structures

- struct ivm_in_get_measurement_t
- struct ivm_out_get_measurement_t
- struct ivm_get_identity_information_t
- struct ivm_start_autocalibration_t
- struct ivm_get_status_t
- struct ivm_check_measurement_status_t
- struct ivm_in_get_measurement_raw_t
- struct ivm_out_get_measurement_raw_t
- struct ivm_get_device_rank_t
- struct ivm_measurement_settings_t
- struct ivm_calibration_settings_t

Macros

- #define **IVM_BUILDER_VERSION_MAJOR** 0
- #define **IVM_BUILDER_VERSION_MINOR** 7
- #define **IVM_BUILDER_VERSION_BUGFIX** 2
- #define **IVM_BUILDER_VERSION_SUFFIX** ""
- #define **IVM_BUILDER_VERSION** "0.7.2"
- #define **IVM_URPC_API_EXPORT** __attribute__((visibility("default")))
- #define **IVM_URPC_CALLING_CONVENTION**
- #define **device_undefined** (-1)
- #define **result_ok** 0
- #define **result_error** (-1)
- #define **result_not_implemented** (-2)
- #define **result_value_error** (-3)
- #define **result_nodevice** (-4)
- #define IVM_FRAME_SIZE 0x19
- #define IVM_CALIBRATION_OK 0x0
- #define IVM_STATUS_OK 0x0
- #define IVM_MEASUREMENT_NOT_COMPLETE 0x0
- #define IVM_MEASUREMENT_COMPLETE 0x1
- #define IVM_RANKING_NOT_SUPPORTED 0x0
- #define IVM_RANK_PRIMARY 0x1
- #define IVM_MIN_NUMBER_POINTS 0xa
- #define IVM_MAX_NUMBER_POINTS 0x3e8
- #define IVM_CURRENT_SENSE_MODE_ISOLATED 0x0
- #define IVM_CURRENT_SENSE_MODE_I_LOW 0x1
- #define IVM_CURRENT_SENSE_MODE_I_MID 0x2
- #define IVM_CURRENT_SENSE_MODE_I_HIGH 0x3
- #define IVM_OUT_MODE_PROBE_SIGNAL_CONTINUOUS 0x0
- #define IVM_OUT_MODE_GROUNDED_CONTINUOUS 0x1
- #define IVM_OUT_MODE_PROBE_SIGNAL_WITH_GROUNDING 0x2

Logging level

- #define LOGLEVEL_ERROR 0x01
- #define LOGLEVEL_WARNING 0x02
- #define LOGLEVEL_INFO 0x03
- #define LOGLEVEL_DEBUG 0x04

Typedefs

- typedef int **device_t**
- typedef int **result_t**
- typedef void(IVM_URPC_CALLING_CONVENTION ∗ ivm_logging_callback_t) (int loglevel, const wchar_t ∗message, void ∗user_data)

Functions

- IVM_URPC_API_EXPORT void IVM_URPC_CALLING_CONVENTION ivm_logging_callback_stderr_wide (int loglevel, const wchar_t ∗message, void ∗user_data)
- IVM_URPC_API_EXPORT void IVM_URPC_CALLING_CONVENTION ivm_logging_callback_stderr_narrow (int loglevel, const wchar_t ∗message, void ∗user_data)
- IVM_URPC_API_EXPORT void IVM_URPC_CALLING_CONVENTION ivm_set_logging_callback (ivm_logging_callback_t cb, void ∗data)
- IVM_URPC_API_EXPORT device_t IVM_URPC_CALLING_CONVENTION ivm_open_device (const char ∗uri)
- IVM_URPC_API_EXPORT device_t IVM_URPC_CALLING_CONVENTION ivm_libversion (char ∗lib_version)
- IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_get_measurement (device_t handle, ivm_in_get_measurement_t ∗input, ivm_out_get_measurement_t ∗output)
- IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_get_identity_information (device_t handle, ivm_get_identity_information_t ∗output)
- IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_start_autocalibration (device_t handle, ivm_start_autocalibration_t ∗output)
- IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_get_status (device_t handle, ivm_get_status_t ∗output)
- IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_start_measurement (device_t handle)
- IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_check_measurement_status (device_t handle, ivm_check_measurement_status_t ∗output)
- IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_get_measurement_raw (device_t handle, ivm_in_get_measurement_raw_t ∗input, ivm_out_get_measurement_raw_t ∗output)
- IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_get_device_rank (device_t handle, ivm_get_device_rank_t ∗output)
- IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_get_measurement_settings (device_t handle, ivm_measurement_settings_t ∗output)
- IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_set_measurement_settings (device_t handle, ivm_measurement_settings_t ∗input)
- IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_get_calibration_settings (device_t handle, ivm_calibration_settings_t ∗output)
- IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_set_calibration_settings (device_t handle, ivm_calibration_settings_t ∗input)
- IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_close_device (device_t ∗handle_ptr)
- IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_get_profile (device_t handle, char ∗∗buffer, void ∗(∗allocate)(size_t))
- IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_set_profile (device_t handle, char ∗buffer)

2.1.1 Detailed Description

ivm API

2.1.2 Data Structure Documentation

2.1.2.1 struct ivm_in_get_measurement_t

Data Fields

uint16_t	FrameNumber	Number of the requested frame. Frame numbering starts from 0.

2.1.2.2 struct ivm_out_get_measurement_t

Data Fields

float	Current[25]	Array of currents (current coordinates of the IV-curve) within the requested frame. Units: mA.
float	Voltage[25]	Array of voltages (voltage coordinates of the IV-curve) within the requested frame. Units: Volts.

2.1.2.3 struct ivm_get_identity_information_t

Data Fields

uint16_t	BootloaderBugfix	Bootloader release version number.
uint8_t	BootloaderMajor	Bootloader major version number.
uint8_t	BootloaderMinor	Bootloader minor version number.
uint8_t	ControllerName[16]	User controller name. This name can be set by user via additional command. Some devices may not support custom controller name setup.
uint16_t	FirmwareBugfix	Firmware release version number.
uint8_t	FirmwareMajor	Firmware major version number.
uint8_t	FirmwareMinor	Firmware minor version number.
uint16_t	HardwareBugfix	Patch number of the hardware version.
uint8_t	HardwareMajor	The major number of the hardware version.
uint8_t	HardwareMinor	Minor number of the hardware version.
uint8_t	Manufacturer[16]	Manufacturer name. The name is set by the manufacturer.
uint8_t	ProductName[16]	Product name. The name is set by the manufacturer.
uint8_t	Reserved[8]	Software should not rely on the value of this field. To provide compatibility with future products the value of this field shouldn't be modified.
uint32_t	SerialNumber	Device serial number.

2.1.2.4 struct ivm_start_autocalibration_t

Data Fields

uint8_t	Result	Calibration result code. In case of successful calibration Result is 0 (CALIBRATION_OK). In case the Result is not 0, some errors happened during calibration process. If you need detailed Result codes description for your device please contact the manufacturer.

2.1.2.5 struct ivm_get_status_t

Data Fields

uint8_t	Reserved[48]	Software should not rely on the value of this field. To provide compatibility with future products the value of this field shouldn't be modified.

Data Fields

uint32_t	Status	Device general status code. Status should be 0 (STATUS_OK) during normal operation. If you need detailed Status codes description for your device please contact the device manufacturer.
int16_t	Temp	Temperature in tenths of degrees C. Some devices may not support the temperature measurements. In this case 0 will be returned.

2.1.2.6 struct ivm_check_measurement_status_t

Data Fields

uint8_t	ReadyStatus	ReadyStatus indicates whether the measurement is finished or not. Possible values: 0 (MEASUREMENT_NOT_COMPLETE) - the measurement is not complete and the measured data is not ready, 1 (MEASUREMENT_COMPLETE) - the measurement is complete and the measured data is ready to be read.
uint8_t	Reserved[15]	Software should not rely on the value of this field. To provide compatibility with future products the value of this field shouldn't be modified.

2.1.2.7 struct ivm_in_get_measurement_raw_t

Data Fields

uint16_t	Frame	Number of the requested frame. Frame numbering starts from 0.

2.1.2.8 struct ivm_out_get_measurement_raw_t

Data Fields

uint16_t	ADCHighCode[25]	ADC High codes within the requested frame.
uint16_t	ADCLowCode[25]	ADC Low codes within the requested frame.

2.1.2.9 struct ivm_get_device_rank_t

Data Fields

uint32_t	Rank	Device rank. Ranking is started from 1 (RANK_PRIMARY). If 0 (RANKING_NOT_SUPPORTED) returned, the device doesn't support ranking.
uint8_t	Reserved0[128]	Software should not rely on the value of this field. To provide compatibility with future products the value of this field shouldn't be modified.

2.1.2.10 struct ivm_measurement_settings_t

Data Fields

uint32_t	CurrentSensorMode	In different modes current sensor uses different current sense resistors. The higher current the smaller current sense resistor should be used. Mode defines the resolution and noise level. Current sense resistors also limit the maximum current. For preventing damage of sensitive components due to high currents such components should be tested in low current mode. This parameter can take one of the following values: CURRENT_SENSE_MODE_I_HIGH, CURRENT_SENSE_MODE_I_MID, CURRENT_SENSE_MODE_I_LOW. More technical details can be found in user manual.
float	MaxVoltage	Harmonic probe signal voltage amplitude. Units: Volts. This is open circuit voltage. Real voltage drop between probes during the measurement will be smaller due to finite current sensor impedance. The current sensor impedance is determined by the CurrentSensorMode parameter. This parameter can take only the values from fixed set of voltages: 1.2, 3.3, 5, 12. If other value is transferred the closest valid value will be set.
uint32_t	NumberChargePoints	This parameter determines delay added testing component precharge before measurement. This parameter is set as number of samples. The value of the delay in seconds can be calculated as NumberChargePoints ∗ SamplingRate. These precharge points are not presented in measurement results. Warning: some devices don't support precharge.
uint32_t	NumberPoints	Number points in a single curve measurements, also referred as an IV curve length. Normally this parameter matches resolution, which is determined by the SamplingRate / ProbeSignalFrequency ratio. However in some special cases the value of this parameter can be differ from resolution for covering several harmonic probe signal periods. Period of time, covered by the single curve measurement can be calculated as NumberPoints ∗ SamplingFrequency. The value of this parameter should be in range determined by constants MIN_NUMBER_POINTS and MAX_NUMBER_POINTS.
uint8_t	OutputMode	Defines electrical connection of both probe connectors to one of the standard sources. During some operations like automatic ADC calibration output mode can be changed automatically from hardware. Note: both ADCs are permanently connected to corresponding probe connectors. Possible values: 0x0 (OUT_MODE_PROBE_SIGNAL_CONTINUOUS). Upper probe is continuously connected to the probe signal source. Lower probe is continuously connected to the ground through the current sense resistor. This is basic mode for measurements. 0x1 (OUT_MODE_GROUNDED_CONTINUOUS). Both probes are continuously connected to ground. Lower probe is connected to the ground through current sense resistor. 0x2 (OUT_MODE_PROBE_SIGNAL_WITH_GROUNDING). Upper probe is connected to the probe signal source during measurement and automatically grounded when measurements are not conducted. Lower probe is continuously connected to the ground through the current sense resistor.
float	ProbeSignalFrequency	Harmonic probe signal frequency. Units: Hz. Normally this parameter is set together with sampling rate for providing fixed resolution. More details in SamplingRate parameter description. ProbeSignalFrequency value should be in range [1, 100 000].
uint8_t	Reserved0[4]	Software should not rely on the value of this field. To provide compatibility with future products the value of this field shouldn't be modified.
uint8_t	Reserved1[19]	Software should not rely on the value of this field. To provide compatibility with future products the value of this field shouldn't be modified.

Data Fields

float	SamplingRate	This rate determines the period between subsequent samples in measured curve. Units: samples / second. Normally this parameter is set together with probe signal frequency for providing fixed resolution, which is determined by the SamplingRate / ProbeSignalFrequency ratio. Examples: desired resolution: 100 samples / period; desired probe signal frequency: 10 Hz; in this case required sampling rate is 10 Hz ∗ 100 points / period = 1000 samples / second; in case of probe signal frequency 1 kHz with the same desired resolution the required sampling rate will be 100 KSPS (the value 100 000 in samples / second should be set). The sampling rate value should be in range [1; 2 000 000].

2.1.2.11 struct ivm_calibration_settings_t

Data Fields

float	ADCMult	Scaling factor for the ADC codes to voltage conversion (for linear model). Units: Volt / count.
float	ADCVOffset	Bias for the ADC codes to voltage conversion (for linear model). Units: Volts.
float	CurrentSense1Mult	Scaling factor for the ADC codes to currents conversion (for linear model). Units: mA / count.
float	CurrentSense1Offset	Bias for the ADC codes to currents conversion (for linear model). Units: mA.
float	CurrentSense2Mult	Scaling factor for the ADC codes to currents conversion (for linear model). Units: mA / count.
float	CurrentSense2Offset	Bias for the ADC codes to currents conversion (for linear model). Units: mA.
float	CurrentSense3Mult	Scaling factor for the ADC codes to currents conversion (for linear model). Units: mA / count.
float	CurrentSense3Offset	Bias for the ADC codes to currents conversion (for linear model). Units: mA.
uint8_t	Reserved0[8]	Software should not rely on the value of this field. To provide compatibility with future products the value of this field shouldn't be modified.
uint8_t	Reserved1[72]	Software should not rely on the value of this field. To provide compatibility with future products the value of this field shouldn't be modified.

2.1.3 Macro Definition Documentation

2.1.3.1 IVM_CALIBRATION_OK

```
#define IVM_CALIBRATION_OK 0x0
```

Calibration complete successfully.

2.1.3.2 IVM_CURRENT_SENSE_MODE_I_HIGH

```
#define IVM_CURRENT_SENSE_MODE_I_HIGH 0x3
```

High current mode.

2.1.3.3 IVM_CURRENT_SENSE_MODE_I_LOW

```
#define IVM_CURRENT_SENSE_MODE_I_LOW 0x1
```

Low current mode.

2.1.3.4 IVM_CURRENT_SENSE_MODE_I_MID

```
#define IVM_CURRENT_SENSE_MODE_I_MID 0x2
```

Medium current mode.

2.1.3.5 IVM_CURRENT_SENSE_MODE_ISOLATED

```
#define IVM_CURRENT_SENSE_MODE_ISOLATED 0x0
```

Current sense resistors disconnected.

2.1.3.6 IVM_FRAME_SIZE

```
#define IVM_FRAME_SIZE 0x19
```

IV curve fame size in points. Is used in get_measurement and get_measurement_raw commands.

2.1.3.7 IVM_MAX_NUMBER_POINTS

```
#define IVM_MAX_NUMBER_POINTS 0x3e8
```

Maximum number points in a single measurement.

2.1.3.8 IVM_MEASUREMENT_COMPLETE

```
#define IVM_MEASUREMENT_COMPLETE 0x1
```

The measurement is complete and the measured data is ready to be read.

2.1.3.9 IVM_MEASUREMENT_NOT_COMPLETE

```
#define IVM_MEASUREMENT_NOT_COMPLETE 0x0
```

The measurement is not complete and the measured data is not ready

2.1.3.10 IVM_MIN_NUMBER_POINTS

```
#define IVM_MIN_NUMBER_POINTS 0xa
```

Minimum number points in a single measurement.

2.1.3.11 IVM_OUT_MODE_GROUNDED_CONTINUOUS

```
#define IVM_OUT_MODE_GROUNDED_CONTINUOUS 0x1
```

Both probes are continuously connected to ground. Lower probe is connected to the ground through current sense resistor.

2.1.3.12 IVM_OUT_MODE_PROBE_SIGNAL_CONTINUOUS

```
#define IVM_OUT_MODE_PROBE_SIGNAL_CONTINUOUS 0x0
```

Upper probe is continuously connected to the probe signal source. Lower probe is continuously connected to the ground through the current sense resistor. This is basic mode for measurements.

2.1.3.13 IVM_OUT_MODE_PROBE_SIGNAL_WITH_GROUNDING

```
#define IVM_OUT_MODE_PROBE_SIGNAL_WITH_GROUNDING 0x2
```

Upper probe is connected to the probe signal source during measurement and automatically grounded when measurements are not conducted. Lower probe is continuously connected to the ground through the current sense resistor.

2.1.3.14 IVM_RANK_PRIMARY

```
#define IVM_RANK_PRIMARY 0x1
```

The highest rank (rank of the primary device).

2.1.3.15 IVM_RANKING_NOT_SUPPORTED

```
#define IVM_RANKING_NOT_SUPPORTED 0x0
```

Ranking is not supported.

2.1.3.16 IVM_STATUS_OK

```
#define IVM_STATUS_OK 0x0
```

The device is OK.

2.1.3.17 LOGLEVEL_DEBUG

```
#define LOGLEVEL_DEBUG 0x04
```

Logging level - debug

2.1.3.18 LOGLEVEL_ERROR

```
#define LOGLEVEL_ERROR 0x01
```

Logging level - error

2.1.3.19 LOGLEVEL_INFO

```
#define LOGLEVEL_INFO 0x03
```

Logging level - info

2.1.3.20 LOGLEVEL_WARNING

```
#define LOGLEVEL_WARNING 0x02
```

Logging level - warning

2.1.4 Typedef Documentation

2.1.4.1 ivm_logging_callback_t

```
typedef void(IVM_URPC_CALLING_CONVENTION * ivm_logging_callback_t) (int loglevel, const wchar←
_t *message, void *user_data)
```

Logging callback prototype.

Parameters

loglevel	- A logging level.
message	- A message.

2.1.5 Function Documentation

2.1.5.1 ivm_check_measurement_status()

```
IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_check_measurement_status (
            device_t handle,
            ivm_check_measurement_status_t * output )
```

Return information about the most recent measurement. This status changes during the measurement process. This command should be used to check whether the measurement is complete or not before requesting measured data.

Parameters

in	*handle*	- Device ID, obtained by ivm_open_device() function.
out	*output*	- Device out data.

2.1.5.2 ivm_close_device()

```
IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_close_device (
            device_t * handle_ptr )
```

Close specified device.

Parameters

handle_ptr	- An identifier of device.

2.1.5.3 ivm_get_calibration_settings()

```
IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_get_calibration_settings (
            device_t handle,
            ivm_calibration_settings_t * output )
```

This command provides manual calibration coefficients setup. Normally this command should not be used. Calibration coefficients can be found automatically by executing start_autocalibration command. Device also can operate with default calibration coefficients without any calibration. Manually updated coefficients will be applied for all measurements until the next automatic calibration, manual coefficients setup or device reset (hardware or software).

Parameters

in	*handle*	- Device ID, obtained by ivm_open_device() function.
out	*output*	- Device out data.

2.1.5.4 ivm_get_device_rank()

```
IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_get_device_rank (
            device_t handle,
            ivm_get_device_rank_t * output )
```

Return device rank (id) in complex system with several devices. The rank is set by motherboard. If custom mother board is used, check the main user manual to find out information about rank setup. This information can be found in the section with motherboard connection description.

Parameters

in	*handle*	- Device ID, obtained by ivm_open_device() function.
out	*output*	- Device out data.

2.1.5.5 ivm_get_identity_information()

```
IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_get_identity_information (
            device_t handle,
            ivm_get_identity_information_t * output )
```

Return device identity information. It is useful to find your device in a list of available devices or check software compatibility with current device and firmware version. Compatibility check should be performed using the compatibility table. Please contact manufacturer to obtain the most recent version of the compatibility table.

Parameters

in	*handle*	- Device ID, obtained by ivm_open_device() function.
out	*output*	- Device out data.

2.1.5.6 ivm_get_measurement()

```
IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_get_measurement (
            device_t handle,
            ivm_in_get_measurement_t * input,
            ivm_out_get_measurement_t * output )
```

Return measurement result: part (frame) of IV curve obtained during the most recent measurement. The data is retrieved from the device by frames. To read the whole curve frame by frame call this command with different FrameNumber argument values several times. Frame numbering starts from 0. To calculate the number of frames divide expected curve length (in points) by the FRAME_SIZE. The curve length can be controlled by the get/set_↩ measurement_settings command. The FRAME_SIZE is constant and equals to 25 points. In case the curve length is not the multiple of the FRAME_SIZE the last frame will be incomplete. The software should not relay on the residual points of the last frame. Example: to read curve consisting of 120 points 5 frames should be requested (120 / 25 = 4.8 => 5 frames); this command should be called 5 times with the FrameNumber argument values: 0, 1, 2, 3, 4. In the last frame only first 20 points will be valid. To receive valid data this command should be called when the measurement will be fully completed. To check whether the measurement is completed or not use check_measurement_status command. To launch new measurement use start_measurement command. To plot IV curve merge all frames into two large voltage and current arrays. Then make a plot using voltages as X coordinates and currents as Y coordinates of the curve points.

Parameters

in	*handle*	- Device ID, obtained by ivm_open_device() function.
in	*input*	- Device in data.
out	*output*	- Device out data.

2.1.5.7 ivm_get_measurement_raw()

```
IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_get_measurement_raw (
            device_t handle,
            ivm_in_get_measurement_raw_t * input,
            ivm_out_get_measurement_raw_t * output )
```

Return raw measurement result: part (frame) of IV curve ADC codes. Note: this command should be used for debug or highly customized measurements. Normally measurement results should be retrieved by the get_measurement command. The data is retrieved from the device by frames. To read the whole curve frame by frame call this command with different FrameNumber argument values several times. Frame numbering starts from 0. To calculate the number of frames divide expected curve length (in points) by the FRAME_SIZE. The curve length can be controlled by the get/set_measurement_settings command. The FRAME_SIZE is constant and equals to 25 points. In case the curve length is not the multiple of the FRAME_SIZE the last frame will be incomplete. The software should not relay on the residual points of the last frame. Example: to read curve consisting of 120 points 5 frames should be requested (120 / 25 = 4.8 => 5 frames); this command should be called 5 times with the FrameNumber argument values: 0, 1, 2, 3, 4. In the last frame only first 20 points will be valid. To receive valid data this command should be called when the measurement will be fully completed. To check whether the measurement is completed or not use check_measurement_status command. To launch new measurement use start_measurement command.

Parameters

in	*handle*	- Device ID, obtained by ivm_open_device() function.
in	*input*	- Device in data.
out	*output*	- Device out data.

2.1.5.8 ivm_get_measurement_settings()

```
IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_get_measurement_settings (
            device_t handle,
            ivm_measurement_settings_t * output )
```

Settings for probe signal generator, current and voltage sensors. Updated settings will be applied for all measurements until the next update settings command execution or device reset.

Parameters

in	*handle*	- Device ID, obtained by ivm_open_device() function.
out	*output*	- Device out data.

2.1.5.9 ivm_get_profile()

```
IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_get_profile (
             device_t handle,
             char ** buffer,
             void *(*)(size_t) allocate )
```

Load profile from device.

Parameters

in	*handle*	- Device id.
out	*buffer*	- Pointer to output char∗ buffer. Memory for char∗ pointer must be allocated.
out	*allocate*	- Function for memory allocate.

2.1.5.10 ivm_get_status()

```
IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_get_status (
             device_t handle,
             ivm_get_status_t * output )
```

Return current device status.

Parameters

in	*handle*	- Device ID, obtained by ivm_open_device() function.
out	*output*	- Device out data.

2.1.5.11 ivm_libversion()

```
IVM_URPC_API_EXPORT device_t IVM_URPC_CALLING_CONVENTION ivm_libversion (
             char * lib_version )
```

Get library version.

Parameters

out	*lib_version*	- Library version.

2.1.5.12 ivm_logging_callback_stderr_narrow()

```
IVM_URPC_API_EXPORT void IVM_URPC_CALLING_CONVENTION ivm_logging_callback_stderr_narrow (
            int loglevel,
            const wchar_t * message,
            void * user_data )
```

Simple callback for logging to stderr in narrow (single byte) chars.

Parameters

loglevel	- A logging level.
message	- A message.

2.1.5.13 ivm_logging_callback_stderr_wide()

```
IVM_URPC_API_EXPORT void IVM_URPC_CALLING_CONVENTION ivm_logging_callback_stderr_wide (
            int loglevel,
            const wchar_t * message,
            void * user_data )
```

Simple callback for logging to stderr in wide chars.

Parameters

loglevel	- A logging level.
message	- A message.

2.1.5.14 ivm_open_device()

```
IVM_URPC_API_EXPORT device_t IVM_URPC_CALLING_CONVENTION ivm_open_device (
            const char * uri )
```

Open a device by name *name* and return identifier of the device which can be used in calls.

Parameters

in	*name*	- A device name. Device name has form "com:port" or "xi-net://host/serial". In case of USB-COM port the "port" is the OS device uri. For example "com:\\.\COM3" in Windows or "com:///dev/ttyACM34" in Linux/Mac. In case of network device the "host" is an IPv4 address or fully qualified domain uri (FQDN), "serial" is the device serial number in hexadecimal system. For example "xi-net://192.168.0.1/00001234" or "xi-net://hostname.com/89ABCDEF". Note: only one program may use COM-device in same time. If errors occur when opening device, you need to make sure that the COM port is in the system and device is not currently used by other programs.

2.1.5.15 ivm_set_calibration_settings()

```
IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_set_calibration_settings (
             device_t handle,
             ivm_calibration_settings_t * input )
```

This command provides manual calibration coefficients setup. Normally this command should not be used. Calibration coefficients can be found automatically by executing start_autocalibration command. Device also can operate with default calibration coefficients without any calibration. Manually updated coefficients will be applied for all measurements until the next automatic calibration, manual coefficients setup or device reset (hardware or software).

Parameters

in	*handle*	- Device ID, obtained by ivm_open_device() function.
in	*input*	- Device in data.

2.1.5.16 ivm_set_logging_callback()

```
IVM_URPC_API_EXPORT void IVM_URPC_CALLING_CONVENTION ivm_set_logging_callback (
             ivm_logging_callback_t cb,
             void * data )
```

Sets a logging callback. Passing NULL disables logging.

Parameters

logging_callback	a callback for log messages

2.1.5.17 ivm_set_measurement_settings()

```
IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_set_measurement_settings (
             device_t handle,
             ivm_measurement_settings_t * input )
```

Settings for probe signal generator, current and voltage sensors. Updated settings will be applied for all measurements until the next update settings command execution or device reset.

Parameters

in	*handle*	- Device ID, obtained by ivm_open_device() function.
in	*input*	- Device in data.

2.1.5.18 ivm_set_profile()

```
IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_set_profile (
            device_t handle,
            char * buffer )
```

Save profile to device

Parameters

in	*handle*	- Device id.
in	*buffer*	- Input char* buffer.

2.1.5.19 ivm_start_autocalibration()

```
IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_start_autocalibration (
            device_t handle,
            ivm_start_autocalibration_t * output )
```

Launch device automatic calibration. During calibration procedure coefficients for conversion raw ADC data to currents and voltages for different modes will be found. To provide optimal conditions probes should be isolated from any external devices or conducting surfaces during calibration procedure (just leave probes without any electrical contact). This command is launched in blocking mode: response will be received when the calibration will be complete. New calibration coefficients will be applied for all measurements until the next automatic calibration, manual coefficients setup or device reset (hardware or software). Normally calibration should be performed just once after device initialization. In case of thermal drift recalibration can compensate parameters changes. The device firmware also has default calibration coefficients. So measurements can be conducted without automatic calibration in case the default calibration provides accurate results.

Parameters

in	*handle*	- Device ID, obtained by ivm_open_device() function.
out	*output*	- Device out data.

2.1.5.20 ivm_start_measurement()

```
IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_start_measurement (
            device_t handle )
```

Launch single measurement in non-blocking mode. It just send the request to start measurement process (command handling will be finished before the whole measurement process will be completed). Use check_↩measurement_status command to check whether the measurement is finished or not.

Parameters

in	*handle*	- Device ID, obtained by ivm_open_device() function.

Index