EyePoint IVM API

1.0.2

Создано системой Doxygen 1.8.13

Содержание

1 Список файлов

1.1 Файлы

Полный список документированных файлов.

2 Файлы

2.1 Файл ivm.h

ivm API

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

Структуры данных

- 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

Макросы

- #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

Уровень логирования

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

Определения типов

- 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)

Функции

- 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 Подробное описание

ivm API

2.1.2 Структуры данных

2.1.2.1 struct ivm_in_get_measurement_t

Поля структур

uint16_t	FrameNumber	Номер запрашиваемого кадра. Нумерация кадров начинается с 0.

2.1.2.2 struct ivm_out_get_measurement_t

Поля структур

float	Current[25]	Массив токов (координат ВАХ по току) в пределах запрашиваемого кадра. Единицы измерения: мА.
float	Voltage[25]	Массив напряжений (координат ВАХ по напряжению) в пределах запрашиваемого кадра. Единицы измерения: Вольты.

2.1.2.3 struct ivm_get_identity_information_t

Поля структур

uint16_t	BootloaderBugfix	Номер релиза версии загрузчика.
uint8_t	BootloaderMajor	Основной номер версии загрузчика.
uint8_t	BootloaderMinor	Второстепенный номер версии загрузчика.
uint8_t	ControllerName[16]	Пользовательское имя контроллера. Может быть установлено пользователем с помощью отдельной команды. Некоторые устройства могут не поддерживать установку пользовательского имени контроллера.
uint16_t	FirmwareBugfix	Номер релиза версии прошивки.
uint8_t	FirmwareMajor	Основной номер версии прошивки.
uint8_t	FirmwareMinor	Второстепенный номер версии прошивки.
uint16_t	HardwareBugfix	Номер правок этой версии устройства.
uint8_t	HardwareMajor	Основной номер версии устройства.
uint8_t	HardwareMinor	Второстепенный номер версии устройства.
uint8_t	Manufacturer[16]	Имя производителя. Устанавливается производителем.
uint8_t	ProductName[16]	Название продукта. Устанавливается производителем.
uint8_t	Reserved[8]	Значение данного поля не должно использоваться в прикладном ПО. Для обеспечения совместимости с другими устройствами не изменяйте значение этого поля.
uint32_t	SerialNumber	Серийный номер устройства.

2.1.2.4 struct ivm_start_autocalibration_t

Поля структур

uint8_t	Result	Код результата калибровки. В случае успешного завершения калибровки Result принимает значение 0 (CALIBRATION_OK). В случае если значение Result отлично от нуля, в процессе калибровки возникли ошибки. Если Вам требуется подробное описание возможных кодов результата для Вашего устройства, пожалуйста, свяжитесь с производителем.

2.1.2.5 struct ivm_get_status_t

Поля структур

uint8_t	Reserved[48]	Значение данного поля не должно использоваться в прикладном ПО. Для обеспечения совместимости с другими устройствами не изменяйте значение этого поля.
uint32_t	Status	Общий код состояния устройства. В нормальном режиме работы значение Status должно быть 0 (STATUS_OK). Если Вам требуется подробное описание возможных значений параметра Status для Вашего устройства, пожалуйста, свяжитесь с производителем.
int16_t	Temp	Температура процессора в десятых долях градусов цельсия. Некоторые устройства не поддерживают измерения температуры. В этом случае будет возвращаться 0.

2.1.2.6 struct ivm_check_measurement_status_t

Поля структур

uint8_t	ReadyStatus	ReadyStatus позволяет узнать завершилось ли измерение. Возможные значения: 0 (MEASUREMENT_NOT_COMPLETE) – измерение не завершено и измеренные данные не готовы, 1 (MEASUREMENT_COMPLETE) – измерение завершено, можно осуществлять чтение измеренных значений.
uint8_t	Reserved[15]	Значение данного поля не должно использоваться в прикладном ПО. Для обеспечения совместимости с другими устройствами не изменяйте значение этого поля.

2.1.2.7 struct ivm_in_get_measurement_raw_t

Поля структур

uint16_t	Frame	Номер запрашиваемого кадра. Нумерация кадров начинается с 0.

2.1.2.8 struct ivm_out_get_measurement_raw_t

Поля структур

uint16_t	ADCHighCode[25]	Коды верхнего АЦП в рамках запрашиваемого кадра.
uint16_t	ADCLowCode[25]	Коды нижнего АЦП в рамках запрашиваемого кадра.

2.1.2.9 struct ivm_get_device_rank_t

Поля структур

uint32_t	Rank	Ранг устройства. Ранжирование начинается с 1 (RANK_PRIMARY). Если получен 0 (RANKING_NOT_SUPPORTED), устройство не поддерживает ранжирование.

Поля структур

uint8_t	Reserved0[128]	Значение данного поля не должно использоваться в прикладном ПО. Для обеспечения совместимости с другими устройствами не изменяйте значение этого поля.

2.1.2.10 struct ivm_measurement_settings_t

Поля структур

uint32_t	CurrentSensorMode	В разных режимах измеритель тока использует токоизмерительные резисторы разного номинала. Чем больше ток, тем меньше должен быть номинал токоизмерительного резистора. Режим определяет разрешение и уровень шумов. Токоизмерительные резисторы также ограничивают максимальный ток. Чтобы предотвратить повреждение чувствительных компонентов из-за протекания больших токов, такие компоненты должны тестироваться в режиме малых токов. Этот параметр может принимать одно из следующих значений: CURRENT_SENSE_MODE_I_HIGH (режим больших токов), CURRENT_SENSE_MODE_I_MID (режим средних токов), CURRENT_SENSE_MODE_I_LOW (режим малых токов). Более подробную техническую информацию можно найти в основном руководстве к прибору.
float	MaxVoltage	Амплитуда напряжения пробного гармонического сигнала. Единицы измерения: Вольты. Это значение может достигаться в режиме холостого хода. Реальное падение напряжения между щупами в процессе измерения будет меньше из-за конечного импеданса измерителя тока. Импеданс измерителя тока определяется значением параметра CurrentSensorMode. Этот параметр может принимать значения только из фиксированного набора напряжений: 1.2, 3.3, 5 и 12. В случае, если передано другое значение будет установлено ближайшее значение из списка допустимых.
uint32_t	NumberChargePoints	Этот параметр определяет задержку добавляемую для предварительной зарядки исследуемого компонента перед измерением. Этот параметр задаётся в виде количества точек. Величина задержки в секундах может быть посчитана как произведение параметров NumberChargePoints $*$ SamplingRate. Точки предварительного заряда в результаты измерений не попадают. Предупреждение: некоторые устройства не поддерживают предварительный заряд.

Поля структур

uint32_t	NumberPoints	Число точек, получаемое в ходе одного измерения кривой, этот параметр также упоминается как длина BAX или длина кривой. Обычно этот параметр совпадает с разрешением, которое определяется отношением SamplingRate / ProbeSignalFrequency. Тем не менее в некоторых случаях, когда требуется измерить несколько периодов пробного сигнала значение данного параметра может отличаться от разрешения. Величина промежутка времени на протяжении которого производится измерение одной кривой может быть посчитана как произведение NumberPoints ∗ SamplingFrequency. MIN_NUMBER_POINTS и MAX_NUMBER_POINTS. Значение данного параметра должно находиться в промежутке определяемом константами MIN_NUMBER_POINTS и MAX_NUMBER_POINTS.
uint8_t	OutputMode	Определяет подключение разъёмов обоих щупов к одному из стандартных источников. В ходе некоторых операций, например, таких как автоматическая калибровка АЦП, режим подключения щупов может автоматически изменяться. Замечание: оба АЦП постоянно подключены к соответствующим разъёмам щупов. Возможные значения: 0 (OUT_MODE_PROBE_SIGNAL_CONTINUOUS). Верхний щуп непрерывно подключен к источнику пробного сигнала. Нижний щуп постоянно подключен к земле через токоизмерительный резистор. Этот режим является базовым для проведения измерений. 1 (OUT_MODE_GROUNDED_CONTINUOUS). Верхний щуп непрерывно подключен к источнику пробного сигнала. Нижний щуп постоянно подключен к земле через токоизмерительный резистор. Этот режим является базовым для проведения измерений. 2 (OUT_MODE_PROBE_SIGNAL_WITH_GROUNDING). Верхний щуп подключен к источнику пробного сигнала в процессе измерения и автоматически заземляется в случаях, когда измерения не производятся. Нижний щуп постоянно подключен к земле через токоизмерительный резистор.
float	ProbeSignalFrequency	Частота пробного гармонического сигнала. Единицы измерения: Гц. Обычно этот параметр устанавливается совместно с частотой дискретизации для обеспечения фиксированного разрешения. Подробности в описании параметра SamplingRarte. Значение параметра ProbeSignalFrequency должно находиться в диапазоне [1, 100 000].
uint8_t	Reserved0[4]	Значение данного поля не должно использоваться в прикладном ПО. Для обеспечения совместимости с другими устройствами не изменяйте значение этого поля.
uint8_t	Reserved1[19]	Значение данного поля не должно использоваться в прикладном ПО. Для обеспечения совместимости с другими устройствами не изменяйте значение этого поля.

Поля структур

float	SamplingRate	Частота дискретизации, определяющая промежуток времени между соседними точками в измеряемое последрвательности. Единицы измерения: измерения секунду. Обычно этот параметр устанавливается совместно с частотой пробного сигнала для обеспечения фиксированного разрешения, которое определяется отношением частоты дискретизации к частоте пробного сигнала (SamplingRate / ProbeSignalFrequency). Примеры: желаемое разрешение: 100 точек на период; желаемая частота пробного сигнала: 10 Гц; в этом случае требуемая частота дискретизации будет 10 Гц $*$ 100 точек на период = 1000 измерений в секунду; в случае частоты пробного сигнала 1 кГц при том же желаемом разрешении частота дискретизации будет 100 000 измерений в секунду. Значение параметра SamplingRate должно находиться в диапазоне [1; 2 000 000].

2.1.2.11 struct ivm_calibration_settings_t

Поля структур

float	ADCMult	Коэффициент масштабирования для пересчёта кодов АЦП в напряжения (в линейной модели). Единицы измерения: Вольт / отсчёт.
float	ADCVOffset	Смещение для пересчёта кодов АЦП в напряжения (в линейной модели). Единицы измерения: Вольты.
float	CurrentSense1Mult	Коэффициент масштабирования для пересчёта кодов АЦП в токи (в линейной модели). Единицы измерения: мА / отсчёт.
float	CurrentSense1Offset	Смещение для пересчёта кодов АЦП в токи (в линейной модели). Единицы измерения: мА.
float	CurrentSense2Mult	Коэффициент масштабирования для пересчёта кодов АЦП в токи (в линейной модели). Единицы измерения: мА / отсчёт.
float	CurrentSense2Offset	Смещение для пересчёта кодов АЦП в токи (в линейной модели). Единицы измерения: мА.
float	CurrentSense3Mult	Коэффициент масштабирования для пересчёта кодов АЦП в токи (в линейной модели). Единицы измерения: мА / отсчёт.
float	CurrentSense3Offset	Смещение для пересчёта кодов АЦП в токи (в линейной модели). Единицы измерения: мА.
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]	Значение данного поля не должно использоваться в прикладном ПО. Для обеспечения совместимости с другими устройствами не изменяйте значение этого поля.

2.1.3 Макросы

2.1.3.1 IVM_CALIBRATION_OK

#define IVM_CALIBRATION_OK 0x0

Калибровка завершена успешно.

2.1.3.2 IVM_CURRENT_SENSE_MODE_I_HIGH

#define IVM_CURRENT_SENSE_MODE_I_HIGH 0x3

Режим больших токов.

2.1.3.3 IVM_CURRENT_SENSE_MODE_I_LOW

#define IVM_CURRENT_SENSE_MODE_I_LOW 0x1

Режим малых токов.

2.1.3.4 IVM_CURRENT_SENSE_MODE_I_MID

#define IVM_CURRENT_SENSE_MODE_I_MID 0x2

Режим средних токов.

2.1.3.5 IVM_CURRENT_SENSE_MODE_ISOLATED

#define IVM_CURRENT_SENSE_MODE_ISOLATED 0x0

Токоизмерительные резисторы отключены.

2.1.3.6 IVM_FRAME_SIZE

#define IVM_FRAME_SIZE 0x19

Размер кадра передачи точек вольтамперной характеристики. Используется в командах get_↩ measurement и get_measurement_raw.

2.1.3.7 IVM_MAX_NUMBER_POINTS

#define IVM_MAX_NUMBER_POINTS 0x3e8

Максимальное количество точек в одном измерении.

2.1.3.8 IVM_MEASUREMENT_COMPLETE

#define IVM_MEASUREMENT_COMPLETE 0x1

Измерение завершено, можно осуществлять чтение измеренных значений.

2.1.3.9 IVM_MEASUREMENT_NOT_COMPLETE

#define IVM_MEASUREMENT_NOT_COMPLETE 0x0

Измерение не завершено и измеренные данные не готовы.

2.1.3.10 IVM_MIN_NUMBER_POINTS

#define IVM_MIN_NUMBER_POINTS 0xa

Минимальное количество точек в одном измерении.

2.1.3.11 IVM_OUT_MODE_GROUNDED_CONTINUOUS

#define IVM_OUT_MODE_GROUNDED_CONTINUOUS 0x1

Верхний щуп непрерывно подключен к источнику пробного сигнала. Нижний щуп постоянно подключен к земле через токоизмерительный резистор. Этот режим является базовым для проведения измерений.

2.1.3.12 IVM_OUT_MODE_PROBE_SIGNAL_CONTINUOUS

#define IVM_OUT_MODE_PROBE_SIGNAL_CONTINUOUS 0x0

Верхний щуп непрерывно подключен к источнику пробного сигнала. Нижний щуп постоянно подключен к земле через токоизмерительный резистор. Этот режим является базовым для проведения измерений.

2.1.3.13 IVM_OUT_MODE_PROBE_SIGNAL_WITH_GROUNDING

#define IVM_OUT_MODE_PROBE_SIGNAL_WITH_GROUNDING 0x2

Верхний щуп подключен к источнику пробного сигнала в процессе измерения и автоматически заземляется в случаях, когда измерения не производятся. Нижний щуп постоянно подключен к земле через токоизмерительный резистор.

2.1.3.14 IVM_RANK_PRIMARY

#define IVM_RANK_PRIMARY 0x1

Наивысший ранг (ранг основного устройства).

2.1.3.15 IVM_RANKING_NOT_SUPPORTED

#define IVM_RANKING_NOT_SUPPORTED 0x0

Ранжирование не поддерживается.

2.1.3.16 IVM_STATUS_OK

#define IVM_STATUS_OK 0x0

Устройство функционирует нормально.

2.1.3.17 LOGLEVEL_DEBUG

#define LOGLEVEL_DEBUG 0x04

Уровень логирования - отладка

2.1.3.18 LOGLEVEL_ERROR

#define LOGLEVEL_ERROR 0x01

Уровень логирования - ошибка

2.1.3.19 LOGLEVEL_INFO

#define LOGLEVEL_INFO 0x03

Уровень логирования - информация

2.1.3.20 LOGLEVEL_WARNING

#define LOGLEVEL_WARNING 0x02

Уровень логирования - предупреждение

2.1.4 Типы

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)

Прототип функции обратного вызова для логирования.

Аргументы

loglevel	- Уровень логирования.
message	- Сообщение.

2.1.5 Функции

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)

Получение информации о последнем измерении. В процессе измерения статус изменяется. Эту команду следует использовать для проверки завершения измерения перед запросом результатов измерений.

Аргументы

in	handle	- Идентификатор устройства, полученный от ivm_open_device().
out	output	- Данные, получаемые с устройства.

2.1.5.2 ivm_close_device()

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

Закрывает устройство.

Аргументы

handle_ptr	- Идентификатор устройства.

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 )
```

Эта команда позволяет контролировать значения калибровочных коэффициентов вручную. В нормальном режиме работы эта команда использоваться не должна. Калибровочные коэффициенты могут быть определены автоматически. Для этого нужно выполнить команду start_autocalibration. Устройство также может работать с калибровочными коэффициентами, установленными по умолчанию. Коэффициенты, обновлённые вручную, будут применяться ко всем измерениям до следующего запуска автоматической калибровки, установки коэффициентов вручную или перезагрузки устройства (программной или аппаратной).

Аргументы

in	handle	- Идентификатор устройства, полученный от ivm_open_device().
out	output	- Данные, получаемые с устройства.

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 )
```

Возвращает ранг (идентификатор) устройств в составной системе, включающей в себя несколько устройств. Ранг (id) задаётся материнской платой. В случае использования нестандартной материнской платы, обратитесь к руководству пользователя для получения информации об установке ранга устройства. Соответствующая информация находится в разделе с описанием подключения устройства к материнской плате.

Аргументы

in	handle	- Идентификатор устройства, полученный от ivm_open_device().
out	output	- Данные, получаемые с устройства.

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 )
```

Возвращает идентификационную информацию об устройстве. Эта информация удобна для поиска нужного устройства среди списка доступных а также проверки совместимости программного обеспечения с текущим устройством и версией прошивки. Проверка совместимости должна производиться с использованием таблицы совместимости. Пожалуйста, свяжитесь с производителем для получения наиболее свежей версии таблицы совместимости.

Аргументы

in	handle	- Идентификатор устройства, полученный от ivm_open_device().
out	output	- Данные, получаемые с устройства.

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 )
```

Чтение результатов измерения: части вольтамперной характеристики, полученной в ходе последнего измерения. Чтение данных с устройства осуществляется по кадрам. Для получения всей кривой эта команда должна быть последовательно вызвана несколько раз с разными значениями аргумента FrameNumber. Нумерация кадров начинается с 0. Для вычисления количества кадров нужно разделить ожидаемую длину кривой (в точках) на размер кадра (FRAME_SIZE). Длина кривой определяется командами get/set_measurement_settings. Размер кадра (константа FRAME_SIZE) фиксирован и равен 25 точкам. В случае, когда длина кривой не кратна размеру кадра последний кадр будет заполнен не до конца. Программное обеспечение не должно использовать значения остаточных точек в последнем кадре. Пример: для получения кривой, содержащей 120 точек требуется запросить 5 кадров (120 / 25 = 4,8 => 5 кадров); эта команда должна быть вызвана 5 раз со значениями аргумента FrameNumber: 0, 1, 2, 3, 4. В последнем кадре только первые 20 точек будут иметь актуальные значения. Для получения актуальных данных эта команда должна вызываться только после полного завершения измерения. Чтобы проверить завершилось ли последнее измерение или нет, можно воспользоваться командой check_measurement_status. Для запуска нового измерения можно воспользоваться командой start_measurement. Чтобы построить график вольтамперной характеристики нужно объединить все полученные кадры в два массива напряжений и токов. Затем нужно построить график, используя напряжения в качестве X-координат, а токи в качестве Y-координат, точек кривой.

Аргументы

in	handle	- Идентификатор устройства, полученный от ivm_open_device().
in	input	- Данные, отправляемые устройству.
out	output	- Данные, получаемые с устройства.

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 )
```

Чтение результатов измерения: части (кадра) кодов АЦП вольтамперной характеристики, полученной в ходе последнего измерения. Замечание: эта команда должна использоваться в отладочных целях или в случае нестандартных измерений. В нормальном режиме для получения результатов измерений должна использоваться команда get_measurement. Чтение данных с устройства осуществляется по кадрам. Для получения всей кривой эта команда должна быть последовательно вызвана несколько раз с разными значениями аргумента FrameNumber. Нумерация кадров начинается с 0. Для вычисления количества кадров нужно разделить ожидаемую длину кривой (в точках) на размер кадра (FRAME_SIZE). Длина кривой определяется командами get/set_measurement_↩ settings. Размер кадра (константа FRAME_SIZE) фиксирован и равен 25 точкам. В случае, когда длина кривой не кратна размеру кадра последний кадр будет заполнен не до конца. Программное обеспечение не должно использовать значения остаточных точек в последнем кадре. Пример: для получения кривой, содержащей 120 точек требуется запросить 5 кадров (120 / 25 = 4,8 => 5 кадров); эта команда должна быть вызвана 5 раз со значениями аргумента FrameNumber: 0, 1, 2, 3, 4. В последнем кадре только первые 20 точек будут иметь актуальные значения. Для получения актуальных данных эта команда должна вызываться только после полного завершения измерения. Чтобы проверить завершилось ли последнее измерение или нет, можно воспользоваться командой check_measurement_status. Для запуска нового измерения можно воспользоваться командой start↩ _measurement.

Аргументы

in	handle	- Идентификатор устройства, полученный от ivm_open_device().
in	input	- Данные, отправляемые устройству.
out	output	- Данные, получаемые с устройства.

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 )
```

Настройки генератора опорного сигнала, измерителей тока и напряжения. Установленные настройки будут применяться ко всем измерениям до следующего вызова команды обновления настроек или перезагрузки устройства.

Аргументы

in	handle	- Идентификатор устройства, полученный от ivm_open_device().
out	output	- Данные, получаемые с устройства.

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 )
```

Загружает профиль с устройства.

Аргументы

in	handle	- Идентификатор устройства.
out	buffer	- Адрес указателя на выходной буфер. Память для указателя на char* должна быть выделена.
out	allocate	- Функция для выделения памяти.

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 )
```

Возвращает информацию о текущем состоянии устройства.

Аргументы

in	handle	- Идентификатор устройства, полученный от ivm_open_device().
out	output	- Данные, получаемые с устройства.

2.1.5.11 ivm_libversion()

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

Версия библиотеки.

Аргументы

out	lib_version	- Версия библиотеки.

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 )
```

Простая функция логирования на stderr в узких (однобайтных) символах.

Аргументы

loglevel	- Уровень логирования.
message	- Сообщение.

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 )
```

Простая функция логирования на stderr в широких символах.

Аргументы

loglevel	- Уровень логирования.
message	- Сообщение.

2.1.5.14 ivm_open_device()

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

Открывает устройство по имени name и возвращает идентификатор устройства.

Аргументы

in	name	- Имя устройства. Имя устройства имеет вид "com:port" или xi-net://host/serial. Для COM устройства "port" это имя устройства в ОС. Например "com:\\.\COM3" (Windows) или "com:///dev/tty/ttyACM34" (Linux/Mac). Для сетевого устройства "host" это IPv4 адрес или полностью определённое имя домена, "serial" это серийный номер устройства в шестнадцатеричной системе. Например "xi-net://192.168.0.1/00001234" или "xi-net://hostname.com/89ABCDEF". Замечание: в один момент времени COM устройство может использоваться только одной программой. Если при открытии устройства возникают ошибки, нужно убедиться, что COM-порт есть в системе и что это устройство в данный момент не используется другими программами

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 )
```

Эта команда позволяет контролировать значения калибровочных коэффициентов вручную. В нормальном режиме работы эта команда использоваться не должна. Калибровочные коэффициенты могут быть определены автоматически. Для этого нужно выполнить команду start_autocalibration. Устройство также может работать с калибровочными коэффициентами, установленными по умолчанию. Коэффициенты, обновлённые вручную, будут применяться ко всем измерениям до следующего запуска автоматической калибровки, установки коэффициентов вручную или перезагрузки устройства (программной или аппаратной).

Аргументы

in	handle	- Идентификатор устройства, полученный от ivm_open_device().
in	input	- Данные, отправляемые устройству.

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 )
```

Устанавливает функцию обратного вызова для логирования. Передача NULL в качестве аргумента отключает логирование.

Аргументы

logging_callback	указатель на функцию обратного вызова

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 )
```

Настройки генератора опорного сигнала, измерителей тока и напряжения. Установленные настройки будут применяться ко всем измерениям до следующего вызова команды обновления настроек или перезагрузки устройства.

Аргументы

in	handle	- Идентификатор устройства, полученный от ivm_open_device().
in	input	- Данные, отправляемые устройству.

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 )
```

Загружает профиль с устройства.

Аргументы

in	handle	- Идентификатор устройства.
in	buffer	- Входной буфер, откуда будет считан профиль.

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 )
```

Запуск автоматической калибровки устройства. В процессе калибровки будут найдены коэффициенты, используемые для пересчёта сырых данных АЦП в токи и напряжения. Для обеспечения оптимальных условий щупы не должны иметь контакта с внешними устройствами или проводящими поверхностями во время калибровки (можно просто оставить щупы «в воздухе» без какого-либо электрического контакта). Команда запускается в блокирующем режиме: ответ будет получен после того, как калибровка будет завершена. Новые калибровочные коэффициенты будут применяться ко всем измерениям до следующего запуска автоматической калибровки, установки коэффициентов вручную или перезагрузки устройства (программной или аппаратной). Обычно калибровку устройства нужно производить всего один раз сразу после инициализации устройства. В случае изменения параметров устройства из-за тепловых воздействий повторная калибровка может компенсировать изменения. В прошивке устройства также есть коэффициенты, которые используются по умолчанию. Таким образом, в случае, если коэффициенты по умолчанию обеспечивают достаточно точные результаты, измерения могут производиться без вызова процедуры автоматической калибровки.

Аргументы

in	handle	- Идентификатор устройства, полученный от ivm_open_device().
out	output	- Данные, получаемые с устройства.

2.1.5.20 ivm_start_measurement()

IVM_URPC_API_EXPORT result_t IVM_URPC_CALLING_CONVENTION ivm_start_measurement (
device_t handle)

Команда осуществляет запуск измерения в неблокирующем режиме. Эта команда только отправляет запрос на запуск измерительного процесса (выполнение команды будет завершено до того, как завершится само измерение). Чтобы проверить завершилось ли запущенное измерение или нет, используйте команду check_measurement_status.

Аргументы

in	handle	- Идентификатор устройства, полученный от ivm_open_device().

Предметный указатель