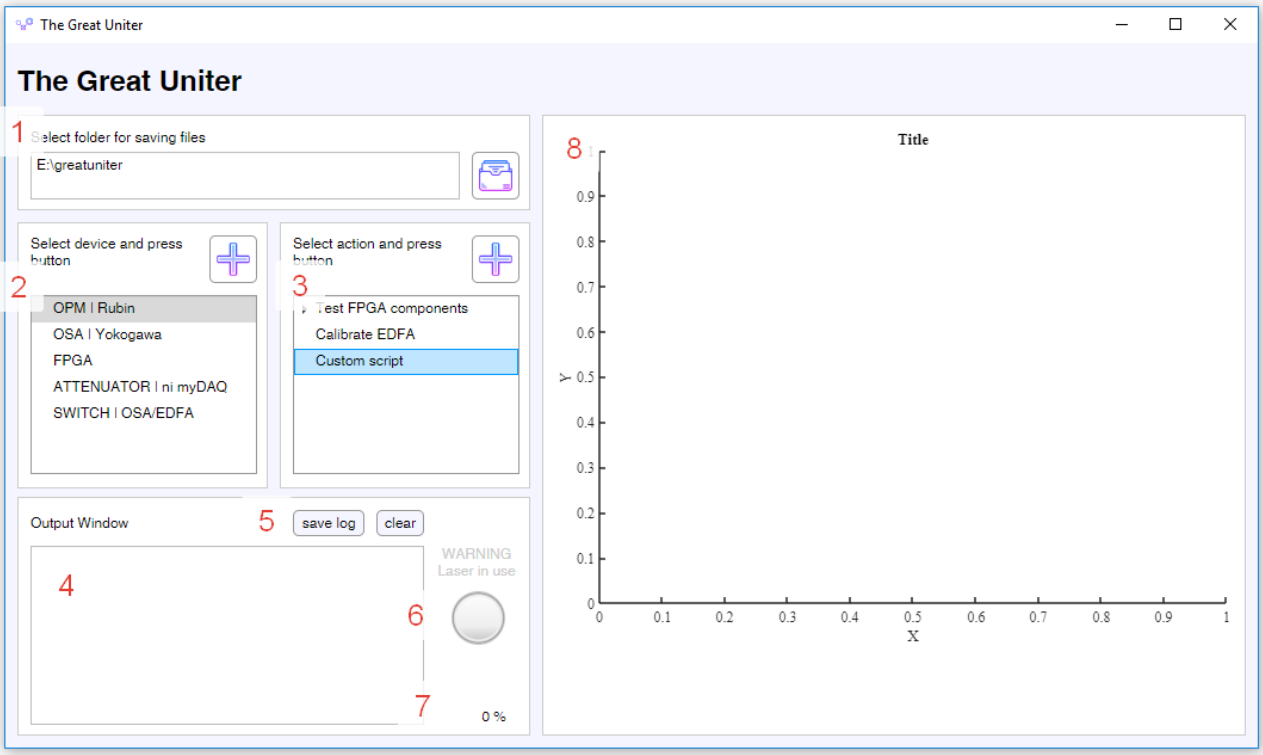
The Great Uniter

Оглавление

Главное окно приложения



1 – Окно выбора папки, куда будут сохраняться все файлы.
2 – Окно выбора устройства. Кликнуть на нужное, нажать +, будет открыто новое окно. Может быть открыто только одно окно для каждого типа устройства, за исключением OPM | Rubin – их может быть несколько.
3 – Окно выбора автоматизированного скрипта. Кликнуть на нужное, нажать +, будет открыто новое окно.
4 – Текстовое поле, в которое будут выводить логи в формате «время: действие».
5 – Логи из текстового поля, которые в нем на данный момент есть, можно скачать, нажав «save log». Кнопка «clear» очищает текстовое поле и CommandWindow в приложении MATLAB.
6 – Лампочка, горит, когда диоды включены. Может не соответствовать действительности, если во время работы программы возникали ошибки. Это связано с тем, что включение и выключение лампочки – это отдельные команды в скрипте, скрипт не проверяет, включен ли сейчас лазер.
7 – Прогресс выполнения какого-либо скрипта в процентах. Может не обновляться, если данная функция не настроена в самом скрипте.
8 – Окно для вывода графика.

Комментарии к следующим разделам

В таблице приведено соответствие названий в инструкции и в главном окне приложения.

OPM | Rubin	Измеритель оптической мощности РУБИН
OSA | Yokogawa	Анализатор спектра Yokogawa
FPGA	ПЛИС с подключением по JTAG
ATTENUATOR | ni myDAQ	Аттенюатор DiCon via NI myDAQ
SWITCH | OSA/EDFA	Переключатель в стенде для проверки одной линии БОУ
SWITCH | OSA/EDFA2lines	Аналогично SWITCH | OSA/EDFA

Для каждого устройства есть своя глава с разделами.
В **возможностях работы с устройством** рассказано о том, что можно делать с устройством при помощи приложения.
Необходимые требования включают в себя список вспомогательного оборудования, версии программного обеспечения, дополнительные файлы.

> **!** Необходимое требование для всего приложения: MATLAB версии 2021b

В **особенностях работы с устройством** даны комментарии по подключению к ПК и внутренних процессах в приложении, подробно расписаны дополнительные действия при работе с устройством, могут быть приведены схемы и таблицы.
Добавление и работа с устройством в приложении знакомят с интерфейсом окна для данного устройства. Обычно интерфейс состоит из двух частей:
1 – Установка начальных параметров и подключение.
2 – Основной функционал устройства.
Основной функционал представлен не только в графическом интерфейсе (пункт 2), но и в виде функций, которые описаны в разделе **создания собственного скрипта с устройством**. Эти функции можно копировать и вставлять в собственные скрипты. Обязательно создать устройство перед тем, как посылать какие-то команды. Команды по созданию отмечены звёздочкой (*).

> **!** Нельзя изменять названия переменных или функций, которые выделены полужирным.

Аттенюатор DiCon via NI myDAQ

Возможности работы с устройством

Выставление аттенюации при помощи регулировки напряжения

Необходимые требования

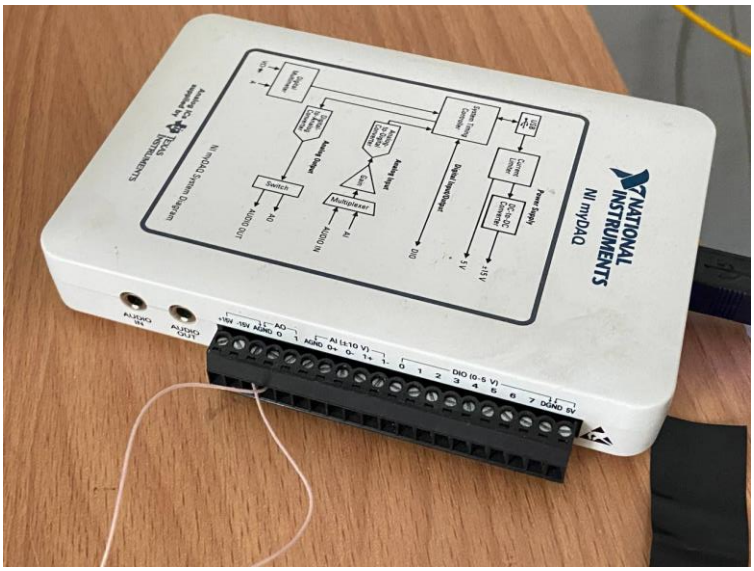
- устройство NI myDAQ и кабель для подключения к ПК
- таблица в формате *.xlsx вида (см пример ниже)
- Curve Fitting Toolbox (рекомендуется Version 3.6)
- Data Acquisition Toolbox (рекомендуется Version 4.4)

Особенности работы с устройством

Для проверки и калибровки БОУ аттенюатор DiCon добавлен в стенды с оптическими ключами. Аттенюатор DiCon с NI myDAQ можно использовать и отдельно от стенда.
Электрические провода аттенюатора (в стенде – выведены) должны быть подключены к NI myDAQ, который в свою очередь подключается к ПК кабелем.

> **!** Провода аттенюатора подключаются так: AO-0 (analog output 0) – «+», AGND – «–».

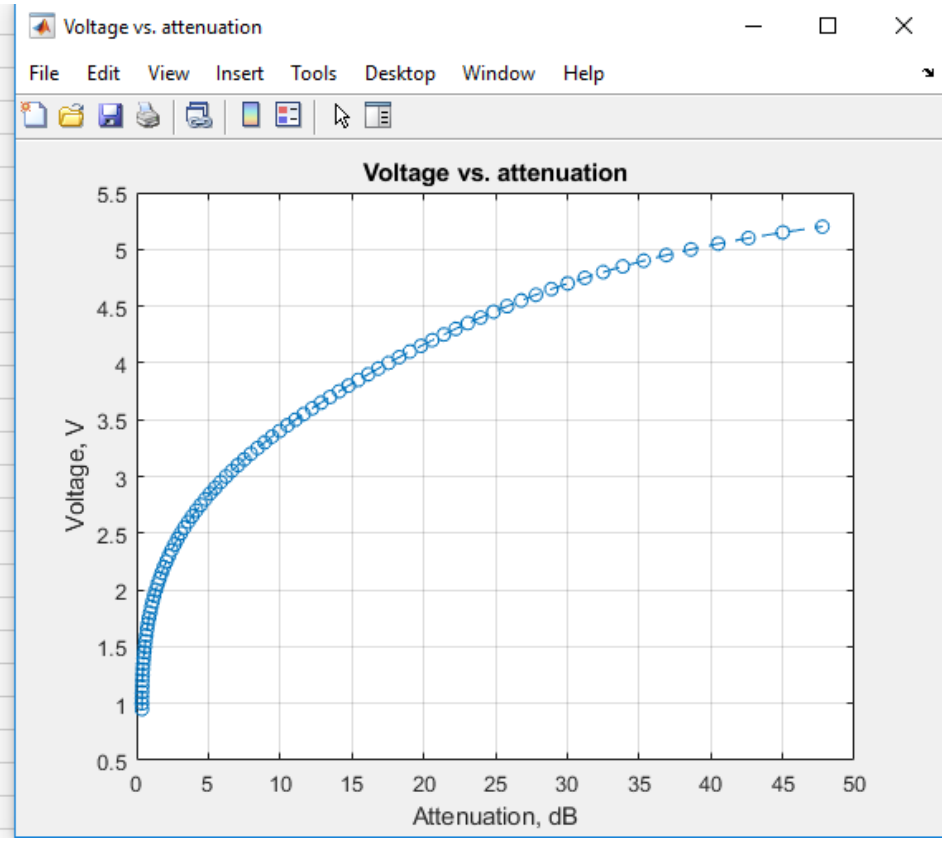
Спецификация NI myDAQ приведена по ссылке.



Таблицу можно создать самостоятельно, данные для заполнения берутся из паспорта на конкретный аттенюатор. Таблица должна быть в формате *.xlsx. В ней должно быть два столбца: затухание в дБ и напряжение в вольтах. По точкам из таблицы строится кривая: ось X – затухание, Y – напряжение.

> **!** Каждому значению затухания должно соответствовать единственное значение напряжения.

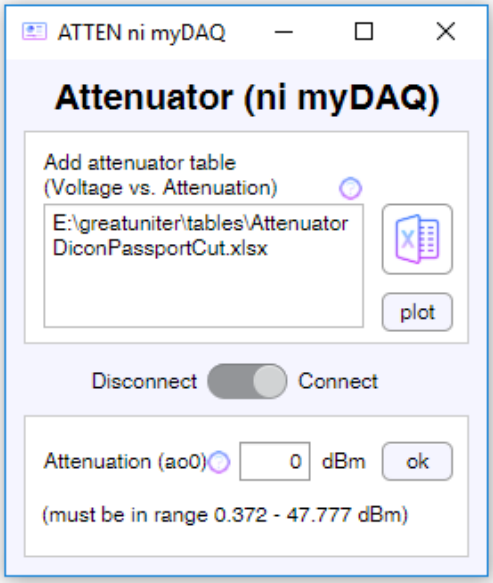
Attenuation_dB	Voltage_V
0,372	0,950
0,374	1,000
0,378	1,050
0,385	1,100
0,396	1,150
0,409	1,200
0,425	1,250
0,446	1,300
0,471	1,350
0,502	1,400
0,536	1,450
0,575	1,500
0,620	1,550
0,671	1,600
0,728	1,650
0,793	1,700
0,865	1,750
0,943	1,800
1,030	1,850
1,126	1,900
1,231	1,950
1,346	2,000
1,468	2,050
1,602	2,100



Работа с устройством в приложении

1. Добавление таблицы. Нажать на кнопку или вручную прописать путь.
2. Проверка, что кривая строится корректно. Нажать **plot**.
3. Подключение устройства. Переключатель в положение **Connect**

После этого станет активной нижняя часть окна. В ней можно прописывать затухание в указанных в таблице пределах и нажимать **ok**.



Создание собственного скрипта с устройством

* Создание объекта

```
device = mainApp.ATTENniMyDaq;
```

Выставление аттенюации по умолчанию. Рекомендуется использовать в начале каждого скрипта.

```
device = setMaxAttenuation(device);
```

Выставление аттенюации, например, 5 дБ.

```
attenuation = 5;
device = setAttenuation(device,attenuation);
```

Запрос последней выставленной аттенюации

```
device.lastAttenuation
```

Удаление объекта

```
device = deleteVirtualObject(device);
```

ПЛИС с подключением по JTAG

Возможности работы с устройством

Чтение и запись значений во временной и постоянной памяти.
Специально для БОУ добавлены функции: включение конкретных диодов, трансиверов и проверка линка на всех трансиверах.

Необходимые требования

- программатор и кабели для подключения к ПК
- файл адресного пространства в формате *.txt вида (см пример ниже)

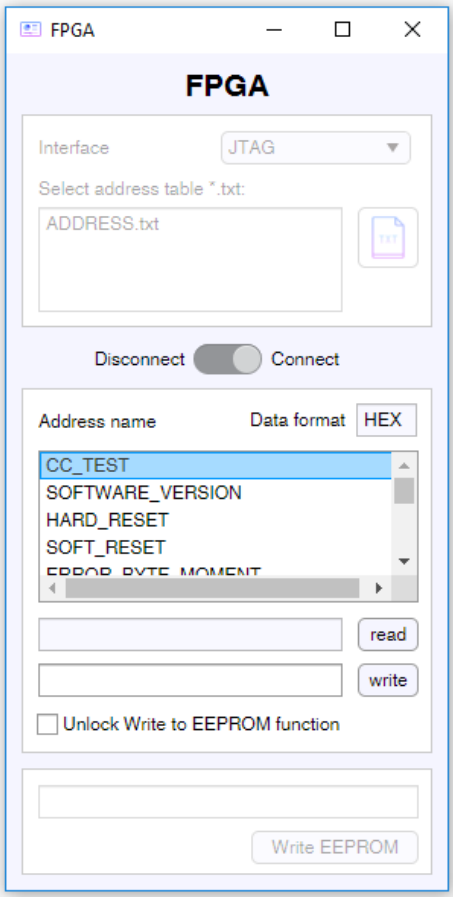
Особенности работы с устройством

ПЛИС подключается к ПК с помощью программатора.

> **!** Работа приложения одновременно с Veryslot протестирована недостаточно. При возникших проблемах попробуйте сначала завершить Veryslot и, возможно, перезапустить MATLAB.

Добавление устройства в приложении

1. Выбор интерфейса подключения. Доступен только **JTAG**.
2. Добавление файла адресного пространства в формате *.txt.
3. Подключение устройства. Переключатель в положение **Connect**.



Работа с устройством в приложении

В списке **Address name** отображаются названия адресов из добавленного txt-файла.
При выборе названия будет определяться его формат и отображаться в поле **Data format**. Данные в полях **read** и **write** должны соответствовать указанному формату.
Как прочитать значение: выбрать адресную ячейку и нажать **read**.
Как записать значение: выбрать ячейку, записать значение в поле **write** и нажать **write**. Если значение будет успешно записано, оно отобразится в поле **read**.

Для записи в EEPROM: поставить галочку, ввести пароль, повторить действия как для записи значения только с полем **write EEPROM** вместо **write**. Если запись произведена успешно, то значение отразится в поле **read**, прочитано оно будет из памяти EEPROM.

Создание собственного скрипта с устройством

* Создание объекта

```
device = mainApp.FPGA;
```

Чтение значения из временной памяти (FLASH) по названию ячейки

```
name = "CC_TEST";
device = readData(device, device.FLASH_MEM, name);
result = device.lastRead;
```

Чтение значения из постоянной памяти (EEPROM) по названию ячейки

```
name = "CC_TEST";
device = readData(device, device.EEPROM_MEM, name);
result = device.lastRead;
```

Запись значения во временную память (FLASH) по названию ячейки

```
name = "CC_TEST";
data = "0"; % must be string
device = writeData(device, device.FLASH_MEM, name, data);
```

Запись значения в постоянную память (EEPROM) по названию ячейки

```
name = "CC_TEST";
data = "0"; % must be string
device = writeData(device, device.EEPROM_MEM,name,data);
```

Также доступна функция, которая включает последовательное выполнение `writeData` и `readData`.

```
mem = device.FLASH_MEM; % или device.EEPROM_MEM
device = writeAndReadData(device, mem, name, data);
result = device.lastRead;
```

Удаление объекта

```
device = deleteVirtualObject(device);
```

Дополнительные функции (актуальны для БОУ)

Включение определенных диодов, например, первого и третьего.

```
binInput = "0101";
% Example: binInput = '0000' => turn off all diodes
% Example: binInput = '0010' => turn on only second diode
% diodes positions are 4321 in binInput

device = setDiodesEn(device, binInput);
```

Включение определенных трансиверов, например, первого и третьего.

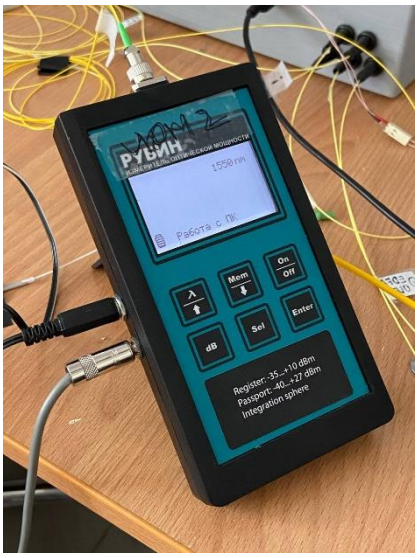
```
binInput = "0101";
% Example: binInput = '0000' => turn off all transceivers
% Example: binInput = '0010' => turn on only second transc
% diodes positions are 4321 in binInput

device = setTransceiversEn(device, binInput);
```

Проверка линка. Ответ придет в виде строки из четырех символов – ноля или единицы, расшифровывается аналогично включению трансиверов.

```
device = checkLink(device);
result = device.currentLinks;
%ex, = '0010' - link at 2nd transc (positions 4321)
```

Измеритель оптической мощности РУБИН

Возможности работы с устройством



Запрос оптической мощности.
Запрос выбранной длины волны.
Выбор длины волны из прописанных в устройстве.
Добавление любого количества измерителей и независимая работа с каждым.

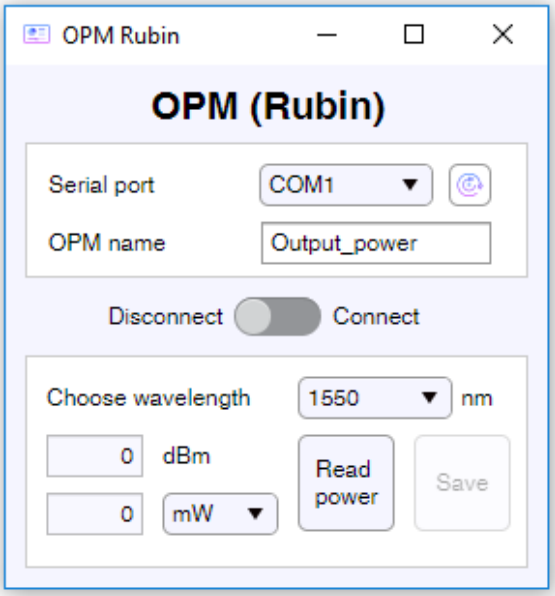
Необходимые требования

- кабели для подключения к ПК

Особенности работы с устройством

Подключение устройства к ПК осуществляется по COM-порту. На устройстве необходимо нажать Enter 3 раза, пока не появится надпись «Работа с ПК».

Добавление устройства в приложении



1. Выбор последовательного порта.
Обновление доступных портов по кнопке.
2. Ввод имени устройства. Особенно важно для калибровки, проверки ФП на плате, собственных скриптов, в остальных случаях можно указать любое.
3. Подключение устройства.
Переключатель в положение **Connect**.

Работа с устройством в приложении

Можно выбрать любую длину волны из выпадающего списка, и она будет выбрана на измерителе (сразу после Connect-а скрипт запрашивает у измерителя список доступных длин волн).
При нажатии **Read power** обновляются поля с оптической мощностью. Поле, значения в котором указаны в Ваттах, автоматически подбирает нужный префикс. Изменить префикс можно, выбрав из выпадающего списка.

Создание собственного скрипта с устройством

* Создание объекта с названием Output_power (значение OPM name при добавлении в приложении)

```
device = mainApp.struct_OPMrubin.Output_power;
```

Запрос длины волны

```
currentWavelength = requestCurrentWavelength(device);
```

Чтение мощности и вывод последней измеренной мощности в дБм

```
device = requestPower(device);
result = device.powerdBm(end);
```

Удаление объекта

```
device = deleteVirtualObject(device);
```

Анализатор спектра Yokogawa

Возможности работы с устройством

Выставление настроек, оффсетов
Запись спектра (делает активной выбранную трассу, записывает на спектроанализаторе)
Чтение и сохранение данных анализа EDFA-NF
Чтение интегральной мощности излучения
Чтение спектра (передает записанные данные на ПК в MATLAB переменную)
Если выполнено чтение спектра, то можно выбрать сохранение спектра на ПК (в формате xlsx).
Если выполнено сохранение, то можно построить полученный график (MATLAB в данном случае не умеет сразу строить график, сначала ему необходимо сохранить файл, а затем прочитать из него).

Необходимые требования

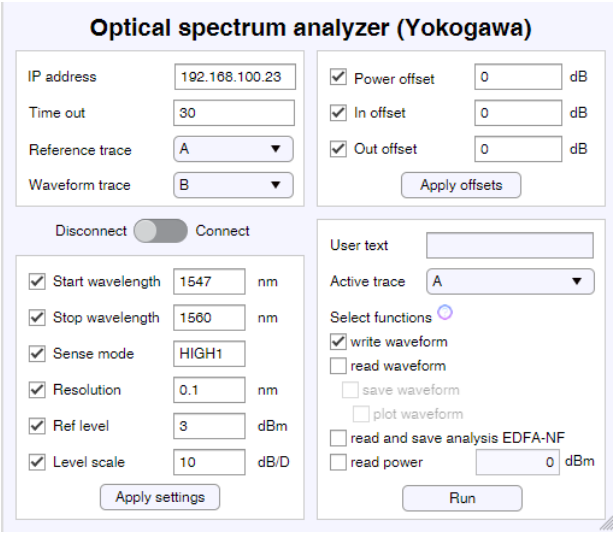
- ПК и спектроанализатор должны быть в одной сети

Особенности работы с устройством

Спектроанализатором нельзя одновременно управлять с ПК и вручную. Если необходимо сделать что-то вручную, можно разорвать соединение (**Disconnect** в приложении), а затем подключиться снова и продолжить работу.

Добавление устройства в приложении

Перепроверить, что все параметры указаны верно и перевести выключатель в положение **Connect.**



Работа с устройством в приложении

1. Окно настроек. Рекомендуется перед работой проверить, что все настройки указаны верно. Если какой-то параметр не нужно применять, уберите отметку. Можно проигнорировать это окно.
2. Окно оффсетов. При необходимости можно выставить нужные оффсеты для спектроанализатора (Можно проигнорировать).
3. Окно основного функционала.
Введите свой текст, он будет добавляться к сохраняемым файлам.
Выберите трассу, на которой вы хотите записать спектр.
Отметьте нужные действия и нажмите **Run**.

Создание собственного скрипта с устройством

* Создание объекта

```
device = mainApp.OSAyokogawa;
```

Запись спектра на трассе A

```
trace = 'A';
writeWaveform(device, trace);
```

Чтение спектра, записанного на трассу A

```
trace = 'A';
device = readWaveform(device, trace);
```

Сохранение последнего прочитанного спектра (невозможно без предыдущего)

```
user_text = 'my_waveform';
device = saveWaveform(device, user_text);
```

Построение последнего прочитанного спектра (невозможно без предыдущего)

```
plotWaveform(device, mainApp.mainWindow);
```

Чтение результата анализа EDFA-NF по всем каналам в виде таблицы

```
trace = 'A';
device = readAnalysisEDFANF(device, trace);
result = device.lastReadAnalysisEDFANF
% пример, как получить столбец NF и найти максимум:
max_NF = max(result{:,'nf'});
```

Сохранение на ПК результата анализа EDFA-NF по всем каналам в виде таблицы

```
user_text = 'my_analysis';
saveAnalysisEDFANF (device, user_text);
```

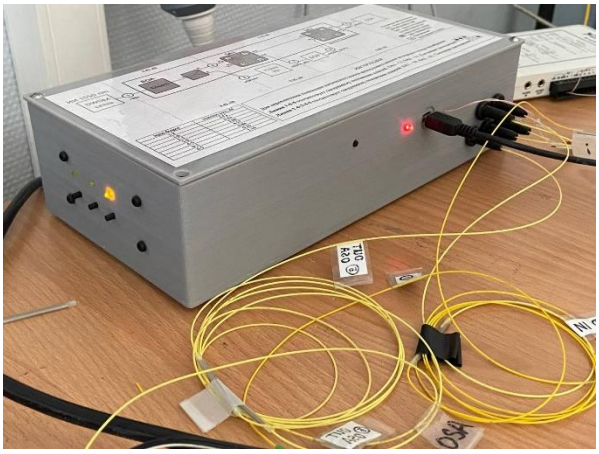
Чтение интегральной мощности в дБм спектра, записанного на трассе A

```
trace = 'A';
power = readPower(device, trace);
```

Удаление объекта

```
device = deleteVirtualObject(device);
```

Переключатель в стенде для проверки одной линии БОУ



Возможности работы с устройством

Управление стендом с оптическими ключами

Необходимые требования

- провод для подключения стенда к ПК

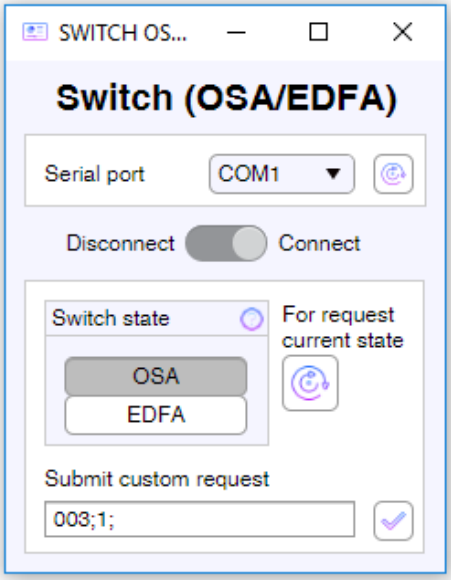
Особенности работы с устройством

Подключение происходит по COM-порту, разъем компьютера – USB. На фото стенд, горящая красная лампочка показывает, что устройство готово к работе.

Добавление устройства в приложении

1. Выбор последовательного порта. При подключении стенда к ПК становится доступен новый порт, обновить список доступных портов можно кнопкой рядом со списком[1].
2. Подключение устройства. Переключатель в положение **Connect**.

Работа с устройством в приложении



Кнопка с подписью «For request current state» обновляет **Switch state**, то есть слева из двух кнопок будет нажата та, в каком положении сейчас ключ.

Изменение положения ключей происходит при выборе другого состояния, то есть, чтобы установить переключатель в положение «на БОУ», нажмите **EDFA**.

Если необходимо отправить свой запрос по COM-порту, это можно сделать в нижней строке.

[1] Теперь не нужно открывать Диспетчер устройств 😊

Создание собственного скрипта с устройством

* Создание объекта

```
device = mainApp.SWITCHosaEdfa;
```

Запрос положения ключей

```
device = requestSwitchState(device);
result = device.switchState;
```

Изменение положения ключей

```
OSAorEDFAstring = "OSA"; % or "EDFA";
device = switchSignalTo(device, OSAorEDFAstring);
```

Удаление объекта

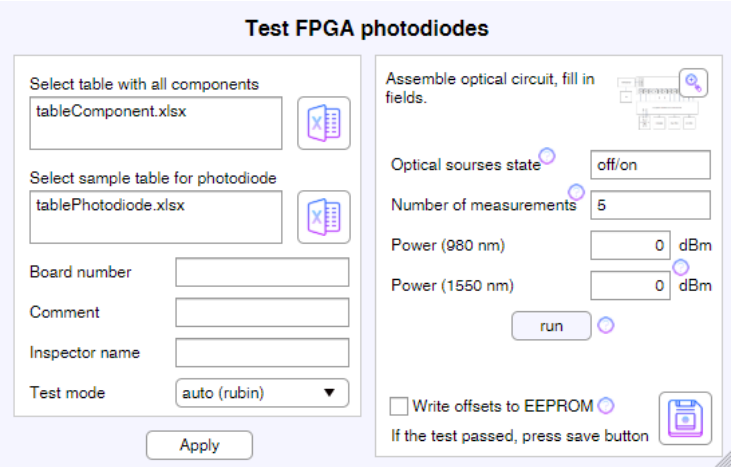
```
device = deleteVirtualObject(device);
```

Проверка электрооптики на плате

В этом разделе будут приведены скрины и даны некоторые пояснения. Основные действия, в т.ч. с оптикой, смотрите в прошлых инструкциях.
Во всех проверках есть две части: ввод данных и проверки. После ввода данных нужно нажать **Apply**, выполнить проверку, сохранить данные и, при необходимости, нажать **Restart** и провести проверку другого компонента без перезапуска приложения/окна.

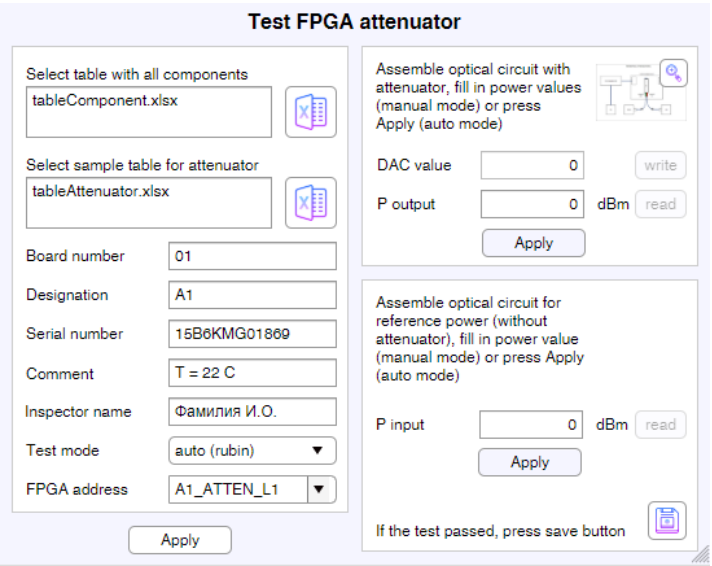
Фотодиоды

Проверка фотодиодов осуществляется с двумя измерителями, назвать их следует так: Power_980 на 980 нм, Power_1550 на 1550 нм.



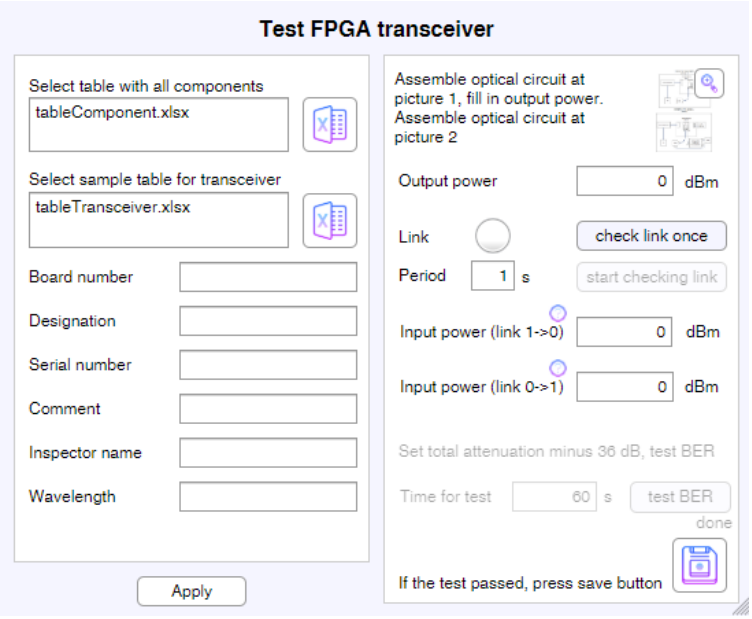
Аттенюатор

Проверка аттенюатора доступна только в авторежиме.



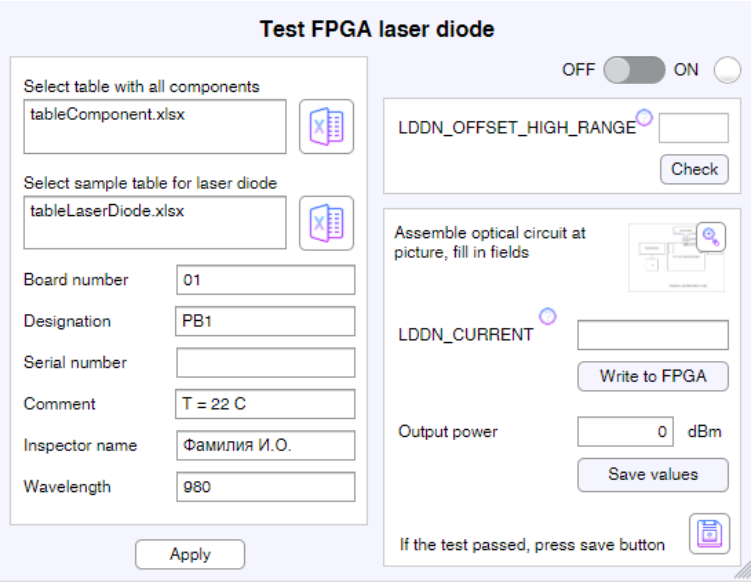
Трансивер

Нет изменений в функциональности по сравнению с прошлой версией.



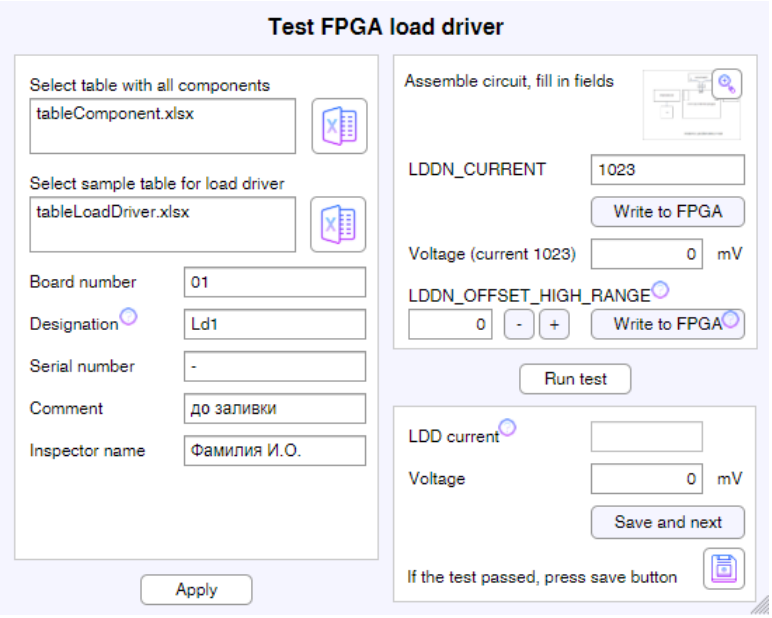
Лазерный диод

Добавлена возможность принудительно включить или выключить лазерные диоды (в правом верхнем углу).
Добавлена возможность ввода разных значений тока и записи всех этих значений в таблицу. Рекомендуется проверять значения 1023, 1024, 1025. Мощность необходимо вносить вручную, можно воспользоваться окном с измерителем в приложении.



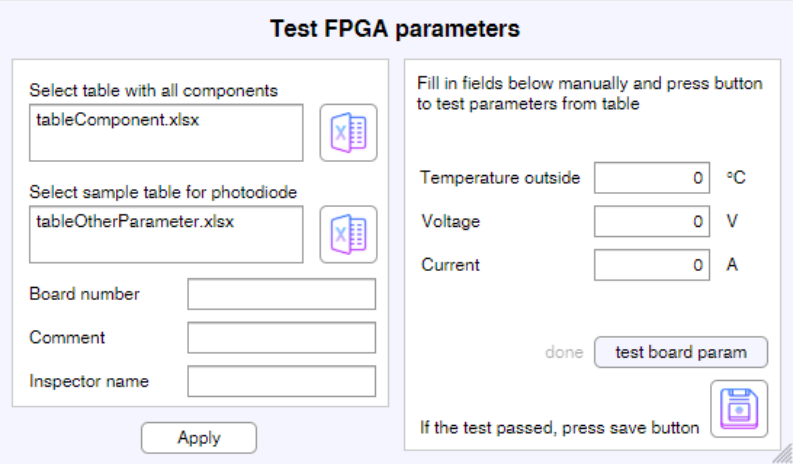
Проверка с имитатором нагрузки

По сравнению с предыдущей версией улучшен интерфейс.



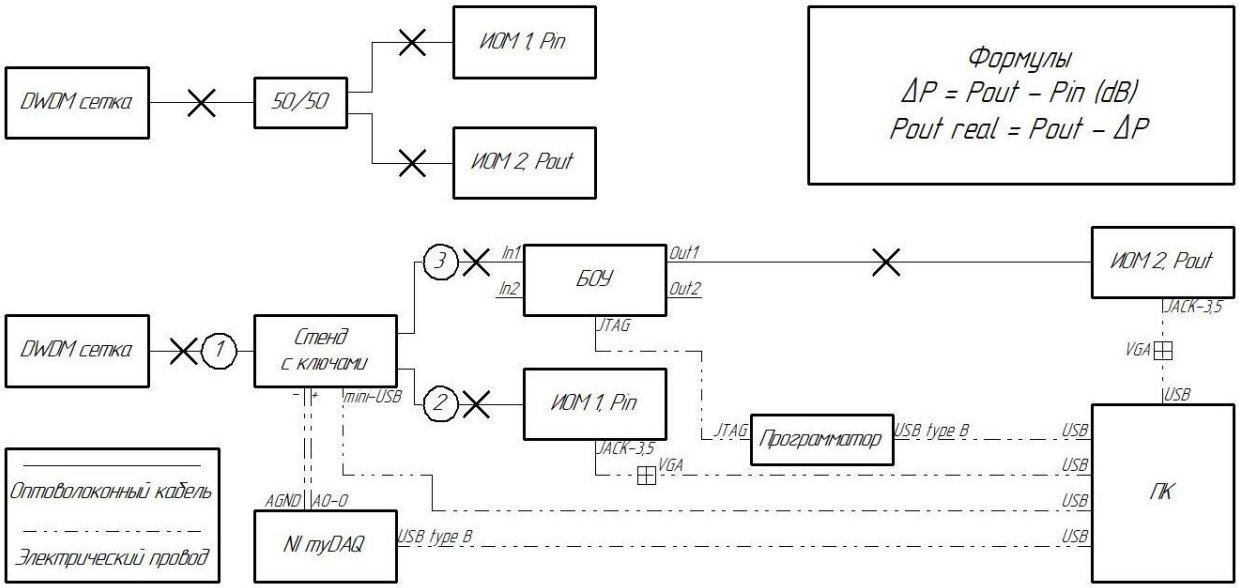
Электрические параметры платы

В таблице **tableOtherParameter.xlsx** указаны референсные значения для каждого параметра. Скрипт при сохранении таблицы проверяет, попадает ли параметр в границы и ставит 1, если попадает.



Калибровка БОУ

! В данной инструкции описаны основные части калибровки БОУ и взаимодействие со скриптом. Подробнее можно посмотреть в редмайне предыдущие версии калибровки или проконсультироваться с коллегами.



1. Определите ΔP между двумя измерителями по схеме.
2. В главном окне приложения добавьте два измерителя с названиями **Pin** и **Pout**. Когда будете добавлять проверьте, что они не перепутаны и дельта вычислена строго по схеме.
3. Добавьте остальные устройства со схемы.
4. Откройте окно калибровки **Calibrate EDFA**.

5. Подготовьте таблицу для калибровки. Она должна соответствовать шаблону **autoCalibration.xlsx**.
6. Заполните все поля и выполните указания. Нажмите **Apply**. Трансиверы на плате будут выключены. Станут активны три окна справа:
    a. получение оффсетов входных и выходных фотоприемников на указанной оптической линии
    b. получение коэффициентов входных фотоприемников на указанной линии
    c. получение коэффициентов выходных фотоприемников на указанной линии
7. При осуществлении пункта 5.a скрипт автоматически переводит ключ стенда в положение OSA. Вы можете дополнительно в этом убедиться, посмотрев на диоды на стенде. Также перед запуском 5.a можно перевести ключ в нужное положение в окне **Switch OSA/EDFA**.
8. В пунктах 5.b и 5.c необходимо заполнить поля (см. вики в редмайне). Отметить нужные действия по завершении калибровки. Далее выбрать какой этап калибровки проводится: получение (**get coeffs**) или проверка (**test coeffs**) коэффициентов. Подробнее об этапах проверки можно посмотреть на вики, в том числе, как осуществить проверку коэффициентов. Рекомендуется записывать коэффициенты во время получения и не записывать их во время теста.
9. По результатам калибровок будут сохранены таблицы с коэффициентами.
10. Для калибровки другой линии нажмите Restart повторите действия с пункта 5.

Создание собственного скрипта

1. Создайте пустой файл в папке, где лежит всё приложение (greatuniter) с названием «custom_» и описанием действий своего скрипта. В этом разделе в качестве примера мы рассмотрим скрипт «**custom_testEDFA_fourLasers.m»**. Из названия следует, что этот скрипт осуществляет проверку БОУ и управляет четырьмя диодами.

2. В начале скрипта создайте раздел Initial parameters, в котором укажите константы, например, с каких трасс спектроанализатора будет идти запись.

```
%% Initial parameters
sampleTablePath = 'C:\greatuniter_test
userText = 'test_';
referenceTrace = 'A';
waveformTrace = 'B';
varNumberToWriteStart = 4;
varNumberToWriteStop = 7;
varNumberToReadStart = 13;
varNumberToReadStop = 27;
```

В **SampleTablePath** задан путь к таблице, из которой будут читаться значения (например, токи лазерных диодов). Таблица выглядит вот так:

Далее в скрипте указаны номера столбцов, для которых будет осуществляться запись или чтение данных из ПЛИС. Для удобства названия адресных ячеек выбраны как названия столбцов. Делать это не обязательно, если адреса вы потом пропишете вручную в коде (при взаимодействии с объектом FPGA). Например, здесь с 4 по 7 столбец указаны параметры токов диодов, которые будут записываться в ПЛИС, а с 13 по 27 – параметры для чтения (температура и фотоприемники).

Таким образом, скрипт получился масштабируемым, можно заменять названия ячеек или добавлять столбцы только в таблице, не изменяя скрипт.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	InputPowerThDBm	InputPowerRealDBm	PauseAfterAttenuationS	LDD1_CURRENT	LDD2_CURRENT	LDD3_CURRENT	LDD4_CURRENT	PauseAfterSetOutputPowerS	OutputPowerRealDBmOSA	MaxNF	MeanGAIN	DeltaGAIN	TEMPERATURE_LD	TEMPERATURE_FPGA	TEMP_CS	ADC1CH5_VD3_Pin_L1_1	ADC1_VD4n_L2
2	-30		5	700	700	700	700	5									
3	-30		5	800	800	800	800	5									
4	-30		5	900	900	900	900	5									
5	-30		5	1000	1000	1000	1000	5									

3. Следующим шагом мы копируем виртуальные версии устройств из основного окна The Great Uniter, куда они будут добавлены при запуске.

```
%% Copy windows from the Great Uniter
OPMrubin_Power_input = mainApp.struct_OPMrubin.Power_input;
OSAyokogawa = mainApp.OSAyokogawa;
FPGA = mainApp.FPGA;
ATTENniMyDaq = mainApp.ATTENniMyDaq;
SWITCHosaEdfa = mainApp.SWITCHosaEdfa;
```

4. Далее идет основной функционал программы – он меняется в зависимости от ваших задач. Команды для общения с устройствами можно посмотреть в разделах для каждого устройства, примеры их применения – в скрипте **custom_testEDFA_fourLasers.m.**

Дополнительные функции в собственном скрипте

Просмотр таблицы в реальном времени

Если в процессе выполнения скрипта заполняется таблица, то просматривать ее можно с помощью этого набора функций.

* Открытие таблицы sampleTable в отдельном окне.

```
statusTable = StatusTable(sampleTable);
```

Обновление или, если она была закрыта, открытие таблицы sampleTable.

```
if ~isvalidFigure(statusTable)
    statusTable = StatusTable(sampleTable);
else
    statusTable = refreshTable(statusTable,sampleTable);
end
```

Вывод логов в Output Window

```
addStrOutput(mainApp.mainWindow,'Hello, world!');
```

Прогресс (целое число в процентах)

```
refreshProgress(mainApp.mainWindow,146);
```

Управление сигнальной лампочкой лазера

Включение предупреждения

```
enableWarningLaser(mainApp.mainWindow);
```

Выключение предупреждения

```
disableWarningLaser(mainApp.mainWindow);
```

Спектры

Удаление всех построенных графиков с координатной плоскости

```
clearAxes(mainApp.mainWindow);
```

Задание названий всего графика, оси X, оси Y.

```
setLabelsAxes(mainApp.mainWindow,'Spectrum(log)','Wavele
ngth, nm','Power, dBm');
```

Задание пределов для отображения графика на плоскости. Если хотите задать предел слева, но не задавать справа (будет автоподстройка), укажите справа **inf**.

```
setLimsAxes(mainApp.mainWindow,[-inf inf],[-80 inf]);
```

Если хотите, чтобы новый график построился поверх того, который уже построен, укажите **on**, иначе **off**.

```
holdAxes(mainApp.mainWindow, 'on');
```

Задание легенды. Число элементов cell-array должно соответствовать количеству графиков. Выполняется после построения.

```
setLegend(mainApp.mainWindow,{'Reference','Waveform'});
```

Запуск собственного скрипта

Чтобы запустить свой скрипт, написанный как указано в разделе «Создание собственного скрипта», необходимо в окне выбора автоматизированного скрипта (стр. 4, номер 3) указать **Custom script**.
Откроется окно, в котором можно добавить путь к своему скрипту и сразу его запустить.

Далее возможны два варианта:

1. Существует заранее написанный скрипт, например, **custom_testEDFA_fourLasers.m.** Тогда вы просто выбираете его и нажимаете стрелочку.
2. У вас нет скрипта, но вы хотите из обычного окна MATLAB программно управлять устройствами из приложения. Тогда необходимо выбрать custom_migrations.m и запустить его. После того, как он успешно выполнится, вы увидите сообщение об этом в Command Window. После этого все устройства из главного приложения будут перенесены в базовое Workspace. Их можно просмотреть, можно отправлять запросы (см. раздел для каждого устройства.

! Все окна The Great Uniter, которые были открыты, должны оставаться открытыми во время работы custom-скрипта.

Возможные ошибки и решения

Если вы не нашли ниже ошибку, которая возникла у вас, свяжитесь с разработчиком или другим ответственным. Подробно опишите свои действия до появления ошибки, скопируйте текст ошибки, приложите скрины или видеозапись экрана, только после этого обращайтесь за помощью.
Если вы смогли самостоятельно установить причину ошибки, выполните действия, указанные выше, и дополните этот раздел – так вы поможете другим пользователям.

Дополнительно

Как получить список тулбоксов (toolbox), установленных в MATLAB

В Command Window в MATLAB написать ver.
В ответ придет список, пример приведён ниже.

```
>> ver
-----------------------------------------------------------------------------------------
MATLAB Version: 9.11.0.1769968 (R2021b)
MATLAB License Number: 968398
Operating System: Майкрософт Windows 10 Pro Version 10.0 (Build 16299)
Java Version: Java 1.8.0_202-b08 with Oracle Corporation Java HotSpot(TM) 64-Bit Server
VM mixed mode
-----------------------------------------------------------------------------------------
MATLAB                                                  Version 9.11        (R2021b)
Simulink                                                Version 10.4        (R2021b)
Curve Fitting Toolbox                                   Version 3.6         (R2021b)
DSP System Toolbox                                      Version 9.13        (R2021b)
Data Acquisition Toolbox                                Version 4.4         (R2021b)
Filter Design HDL Coder                                 Version 3.1.10      (R2021b)
Fixed-Point Designer                                    Version 7.3         (R2021b)
Image Acquisition Toolbox                               Version 6.5         (R2021b)
Image Processing Toolbox                                Version 11.4        (R2021b)
Instrument Control Toolbox                              Version 4.5         (R2021b)
OPC Toolbox                                             Version 5.0.3       (R2021b)
Optimization Toolbox                                    Version 9.2         (R2021b)
Parallel Computing Toolbox                              Version 7.5         (R2021b)
Partial Differential Equation Toolbox                   Version 3.7         (R2021b)
Signal Processing Toolbox                               Version 8.7         (R2021b)
Symbolic Math Toolbox                                   Version 9.0         (R2021b)
```

Функция перевода из дБм в Ватты с разными префиксами

Введите ее в скрипте или в Command Window. В примере 10 дБм переводят в мВт.
Возможные варианты префиксов указаны в комментарии после знаков %

```
dbm = 10;
metricPrefix = 'm';
% 'n' – нано
% 'u' - микро
% 'm' - милли
% ' ' - без префикса, просто Вт
w = dbm2w(dbm, metricPrefix);
```