Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Школа профессионального и академического образования

ДОПУСТИТЬ К ЗАЩИТЕ ПЕРЕД ГЭК

Руководитель образовательной программы
«Радиоэлектронные системы»
_____ Шабунин С.Н.
(подпись)
«18» 06 2021 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Разработка программного обеспечения для *DIY Vector Network Analyzer*

ИРИТ 110401 907 ПЗ

Руководитель: Мительман Ю.Е.
Доцент кафедры Радиоэлектроники и телекоммуникаций,
к.т.н., доцент
Нормоконтролер: Чечеткин В.А.
Студент группы РИМ-291208
Теребов И.А.

Екатеринбург, 2021

РЕФЕРАТ

Пояснительная записка содержит 73 с., 20 рис., 6 источн., 4 прил.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, ВЕКТОРНЫЙ АНАЛИЗАТОР ЦЕПЕЙ, *UVNA*-63, КОЭФФИЦИЕНТ СТОЯЧЕЙ ВОЛНЫ, КОЭФФИЦИЕНТ ОТРАЖЕНИЯ, КОЭФФИЦИЕНТ ПРОХОЖДЕНИЯ

Объектом исследования является прибор *DIY Vector Network Analyzer UVNA*-63.

Целью работы является разработка программного обеспечения для *DIY Vector Network Analyzer UVNA*-63.

В данной работе описана разработка программного обеспечения в программе *Matlab*. Описана работа разработанных функций измерения. Описана работы разработанного графического интерфейса. Доработка методики лабораторных работ.

Актуальность работы обоснована отсутствием программного обеспечения для проведения измерений на векторном анализаторе цепей, который имеется в распоряжении кафедры и может быть задействован в учебном процессе. Разработанное программное обеспечение позволит проводить измерения, производить калибровку аппарата, сохранять полученные данные в различных форматах, отображать полученные данные на графике и устанавливать маркеры на графике.

Область применения является учебная сфера. С помощью разработанного программного обеспечение возможно выполнять лабораторные работы и измерять исследуемые устройства.

Подп. и дата | Взам. инв. № | Инв. № дубл. | Подп. и дата | Инв. № подл.

Ли	Изм.	№ докум.	Подп	Дата
Разраб.		Теребов И.А.		14.06.21
Пров.		Мительман		14.06.21
Т. контр.				
Н. контр.		Чечеткин В.А.		14.06.21
Утв.		Шабунин С.Н.		

ИРИТ 110401 907 ПЗ

Разработка программного обеспечения для DIY Vector Network Analyzer

Лит	Лист	Листов
	2	73

УрФУ ИРИТ-РтФ
Школа профессионального и академического образования

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящей пояснительной записке к выпускной квалификационной работе применяются следующие термины с соответствующими определениями.

Коэффициент стоячей волны – отношение наибольшего значения амплитуды напряжённости электрического или магнитного поля стоячей волны в линии передачи к наименьшему

Векторный анализатор цепей – это электроизмерительный прибор, в котором используется известный сигнал незатухающей гармонической волны для измерения амплитудной и фазовой характеристик радиочастотных и микроволновых компонентов, систем и подсистем как при разработке, так и в производстве

GUI – система средств для взаимодействия пользователя с компьютером, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана (окон, значков, меню, кнопок, списков и т. п.)

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей пояснительной записке к выпускной квалификационной работе применяются следующие сокращения и обозначения.

GUI – graphical user interface

SOLT – short open load thru

ВАЦ – векторный анализатор цепей

КСВ – коэффициент стоячей волны

ПО – программное обеспечение

СВЧ – сверхвысокая частота

ВВЕДЕНИЕ

“Одними из наиболее точных приборов для измерения СВЧ трактов являются векторные анализаторы цепей (ВАЦ). Цепи, которые могут быть измерены с помощью ВАЦ, имеют широкий диапазон применений, начиная от простых устройств, таких как фильтры и усилители, и заканчивая сложными модулями, используемыми в системах телекоммуникаций. ВАЦ представляет собой одну из наиболее сложных частей многоцелевого испытательного оборудования в области высокочастотной и сверхвысокочастотной радиотехники. Он предназначен для измерений комплексных коэффициентов передачи и отражения (элементов матрицы рассеяния) многополюсников” [1].

Целью исследования является разработка программного обеспечения для *DIY Vector Network Analyzer* и доработка методических указаний по лабораторным работам.

Для достижения данной цели были поставлены следующие задачи:

– провести аналитический обзор источников;

– разработать алгоритм работы ПО;

– разработать интерфейс ПО и методику работы с ним;

– доработать методические указания по имеющимся лабораторным работам с использование ВАЦ.

В качестве метрологического обеспечения выступает указанный выше ВАЦ для измерения модуля и фазы коэффициента отражения и передачи, ПО *MATLAB* с университетской лицензией УрФУ.

1 Обзор литературы

На рисунке 1 приведена типичная структурная схема ВАЦ, имеющего один измерительный порт [1]. Одной из основных функций ВАЦ является измерение комплексного коэффициента отражения Г.

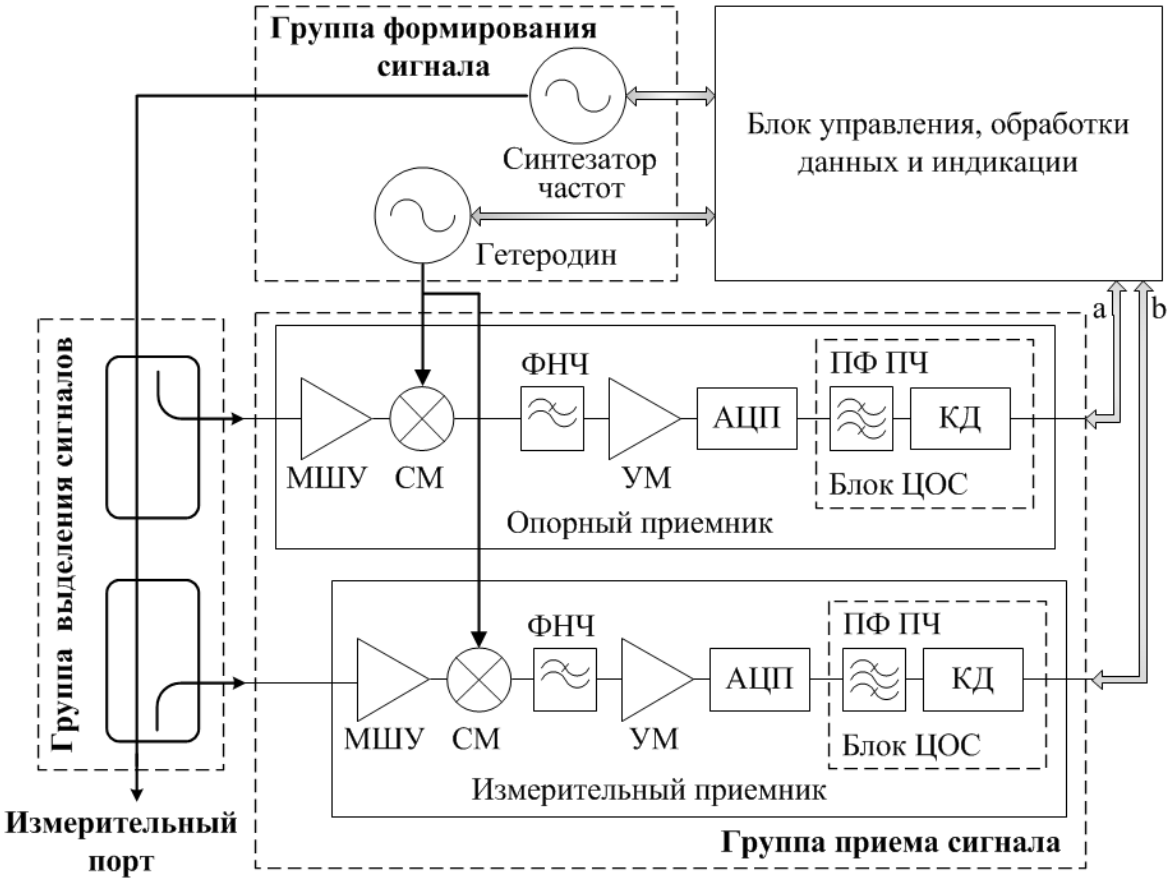


Рисунок 1 – Структурная схема рефлектометра [1]

“В группу формирования сигнала входят два источника: синтезатор частот и гетеродин. Источники синхронизированы по частоте от одного опорного генератора. Синтезатор частот предназначен для формирования зондирующего сигнала в диапазоне рабочих частот ВАЦ. Сигнал гетеродина, смещенный по частоте относительно зондирующего на величину промежуточной частоты, необходим для преобразования (понижения) частоты.

Группа выделения сигналов предназначена для получения падающей и отраженной от исследуемого устройства волн. Для выделения волн

используют направленные ответвители или мосты. Для выделения падающей волны также применяют делители мощности.

Группа приема сигнала состоит из двух идентичных приемников: опорного и измерительного. Для радиотехнических измерений, как правило, приемники строят по супергетеродинной схеме. Они осуществляют преобразование сигналов на более низкую промежуточную частоту, усиление и фильтрацию. В состав приемников входят следующие основные элементы: малошумящий усилитель, смеситель, фильтр нижних частот, аналого-цифровой преобразователь и блок цифровой обработки сигналов:

– малошумящий усилитель предназначен для приема и усиления сигналов с низким уровнем искажений;

– смеситель служит для преобразования на промежуточную частоту;

– фильтр нижних частот предназначен для фильтрации (выделения) сигналов промежуточной частоты на выходе смесителя;

– усилитель мощности предназначен для основного усиления сигналов в приемнике;

– аналогово-цифровой преобразователь служит для преобразования сигнала в цифровой вид;

– блок цифровой обработки сигналов, основными элементами которого являются полосовой фильтр промежуточной частоты и квадратурный демодулятор, предназначен для финальной фильтрации и получения на выходе приемников измерительных сигналов «*a*» и «*b*» в комплексном виде. Сигналы «*a*» и «*b*» в дальнейшем используются для решения измерительной задачи – измерения комплексного коэффициента отражения Г.

К измерительному порту подключаются однопортовые исследуемые устройства. К однопортовым устройствам (двухполюсникам) относятся согласованные и рассогласованные нагрузки, короткозамкнутые нагрузки и нагрузки холостого хода (в том числе меры волнового и полного сопротивлений). Кроме этого, к данному классу устройств можно отнести многопортовые устройства. При этом один из портов такого устройства должен подключаться

к ВАЦ для тестирования, а оставшиеся нагружаться определенным образом” [1].

“Блок управления, обработки данных и индикации отвечает за синхронную работу всех блоков, решает измерительную задачу и отображает результат в выбранных пользователем единицах измерений на экране ВАЦ. Часть функций блока может быть реализована на внешнем персональном компьютере” [1].

Для двухпортовых устройств (четырехполюсников) характеристика отражения от первого порта называется коэффициентом отражения S_{11}, характеристика передачи в прямом направлении называется коэффициентом передачи S_{21}, характеристика передачи в обратном направлении называется коэффициентом передачи S_{12}, и характеристика отражения от второго порта называется коэффициентом отражения S_{22}. Измеряемые параметры представлены на рисунке 2.

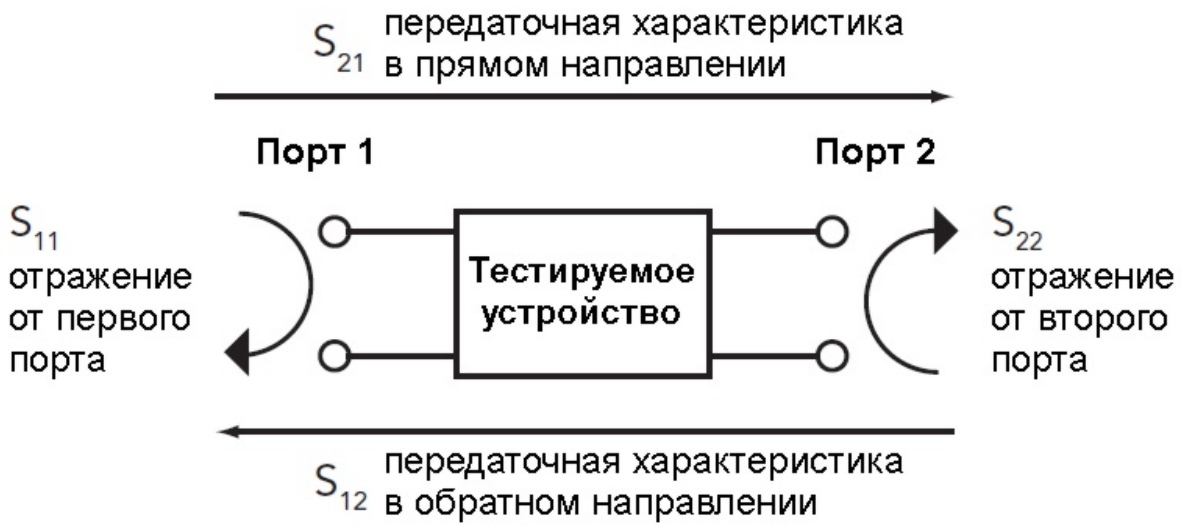


Рисунок 2 – Графическое представление измеряемых параметров [1]

Каждый *S*-параметр содержит амплитудно-частотную и фазо-частотную характеристики тестируемого устройства в соответствующем направлении. Существует много стандартных способов отображения измеренных *S*-параметров на экране ВАЦ. Можно выбирать, в каком виде просматривать результаты: в виде графика КСВ или обратных потерь от частоты, диаграммы Смита,

амплитуды, фазы, вносимого затухания или усиления, групповой задержки и др.

В качестве примера, на рисунке 3 показан экран векторного анализатора *Anritsu VNA Master* серии *MS*20*xxB* с результатами измерения характеристик полосового фильтра.

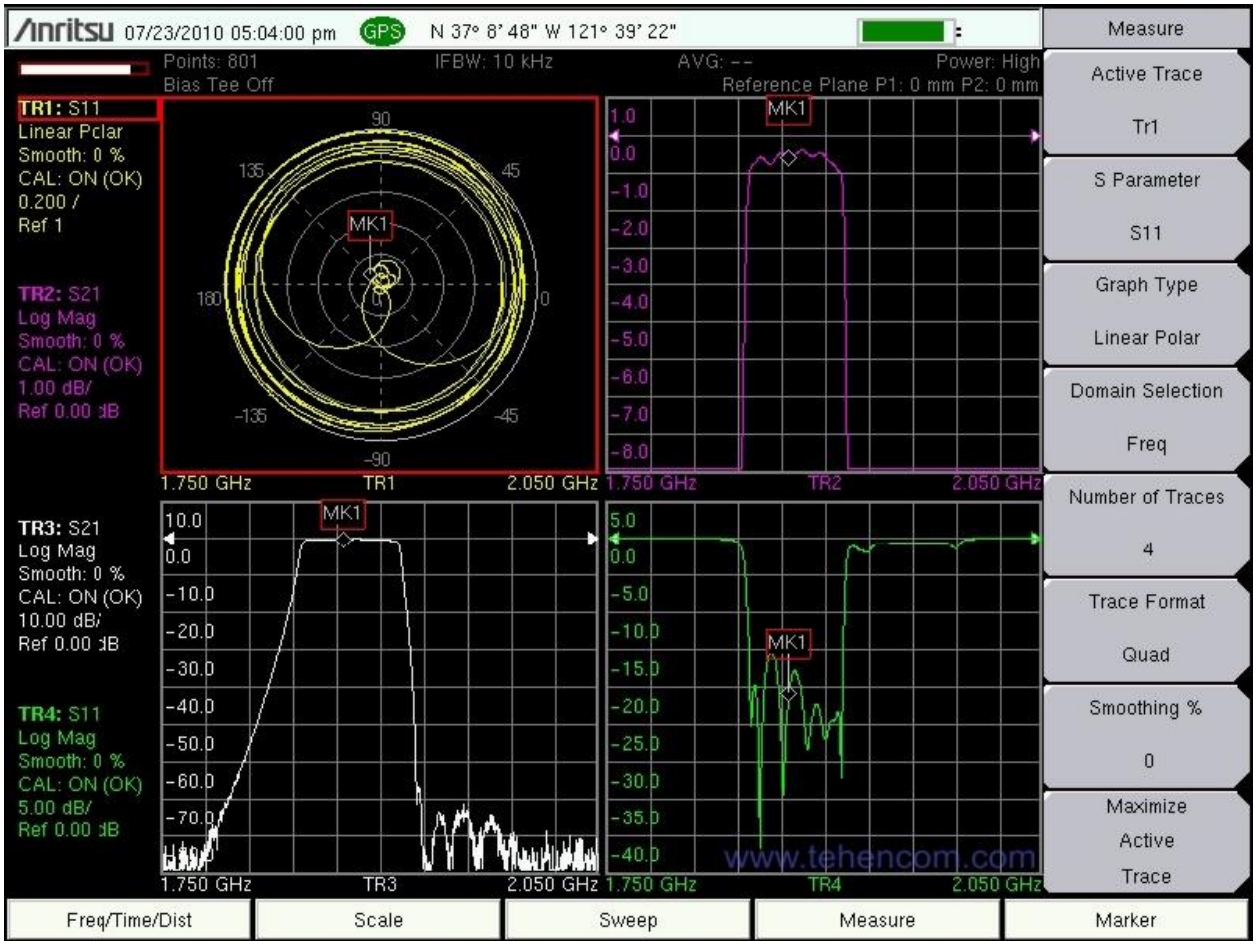


Рисунок 3 – Изображение экрана векторного анализатора цепей [2]

Основные параметры фильтра (S_{11} и S_{21}) представлены на четырёх подробных графиках.

В ходе работы требуется написать программное обеспечение для устройства *DIY Vector Network Analyzer UVNA*-63 [2]. Данное устройство является комплектом для самостоятельной сборки. Оно состоит из набора кабелей, калибровочного набора, платы для обработки полученных данных с портов устройства, различных переходников. Модули данного устройства закрепляются на металлической подложке для хорошей фиксации.

Блок обработки данных имеет шесть портов и питается от *USB* порта компьютера. По этому же интерфейсу происходит прием данных и управление блоком. Вид собранного устройства представлен на рисунке 4.

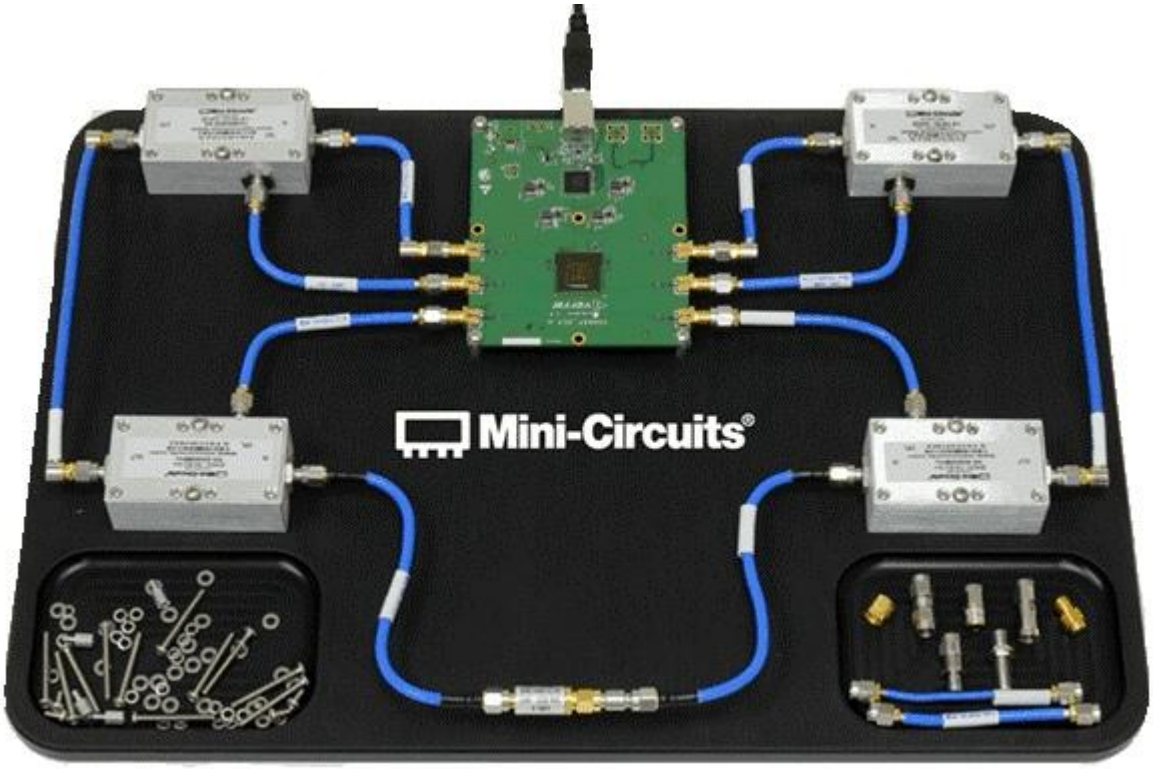
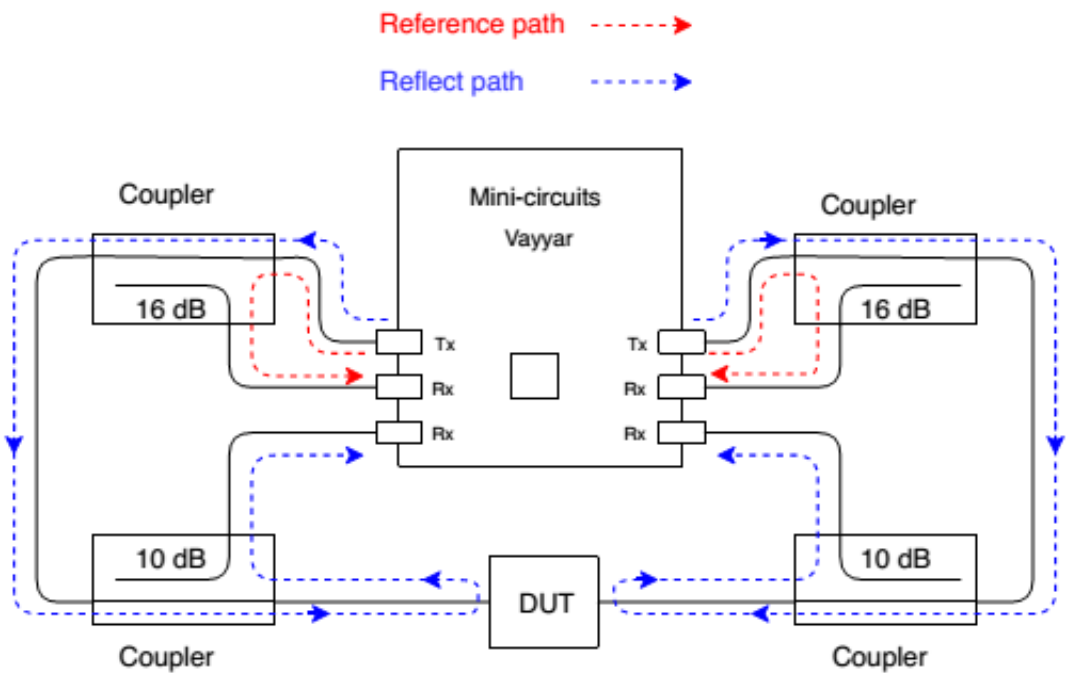


Рисунок 4 – Собранный модульный векторный анализатор цепей [2]

Данный аппарат имеет два порта для измерения, соответственно на нем можно измерять как комплексный коэффициент отражения, так и коэффициент передачи. Структурная схема устройства показана на рисунке 5.



Русунок 5 – Измерительная установка с отображением опорного (красный) и отраженного (синий) путей [3]

При измерении устройств на выход основной платы поступает информация в виде матрицы значений. Полученные данные открываются с помощью программного продукта *MATLAB*. В данной работе необходимо написать программу, которая будет в режиме онлайн преобразовывать данные, полученные с устройства, в графики и диаграммы.

2 Разработка ПО

Программный продукт, разработанный производителем, представляет собой программу *VNAKit.exe*. Интерфейс программы представлен на рисунке 6.

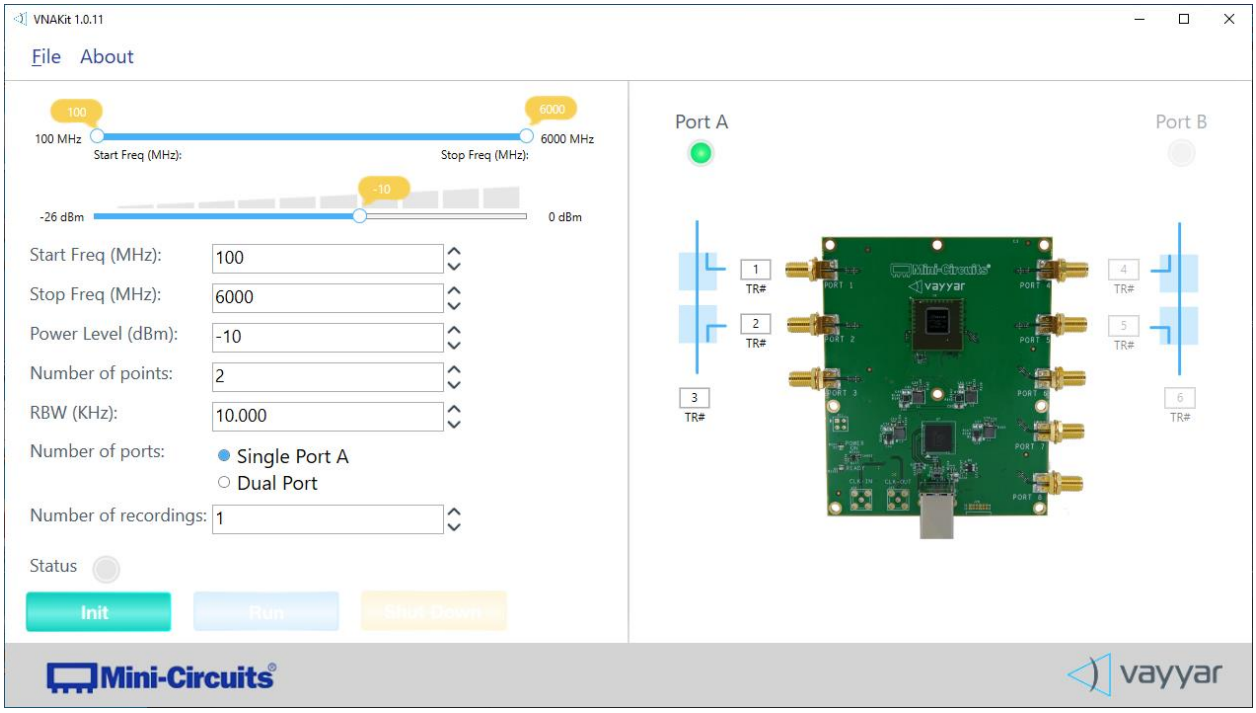


Рисунок 6 – Интерфейс программы *VNAKit* [4]

Программа позволяет установить частоту измерения в пределах от 100 до 6000 МГц, выбрать уровень мощности в пределах от минус 26 до 0 дБм, указать количество измеряемых точек от 2 до 1000 и полосу пропускания от 2 до 140 кГц. Возможно измерение в однопортовом и двухпортовом режимах. Вывод измеренных данных возможен в формате *csv* или *mat*.

В данной программе невозможно отобразить на графике полученные данные и откалибровать полученные значения. Для удобного пользования данным прибором нужно разработать специализированное ПО, чему и посвящена данная работа. Разработка программного обеспечения будет происходить в программного продукте *MATLAB R2020b*.

Для разработки программного обеспечения необходимо, настроить стабильную связь между устройством и компьютером при помощи библиотек, разработанных производителем. В качестве пробного примера был взят пример из библиотек для данного устройства. После доработки функции измерения и коррекции пути до библиотеки была произведена калибровка устройства и измерение в заданном частотном диапазоне.

После разработки всех функций работы с аппаратом и настройки стабильного контакта между устройством и компьютером предстоит разработать графический интерфейс для ВАЦ.

2.1 Алгоритм работы аппарата и ПО

Для разработки функций работы ВАЦ важно знать алгоритм взаимодействия ВАЦ и ПО. Так как ВАЦ питается от *USB*, значит он всегда включен и находится в режиме ожидания соединения или получения команды от ПК. Был разработан алгоритм взаимодействия ВАЦ и ПО. Скорость передачи данных напрямую зависит от скорости работы компьютера и от протокола передачи (*USB* 3.0 или 2.0). При передаче любой команды ВАЦ должен передать данные о выполнении этой команды или о статусе системы.

Для работы с аппаратом нужно запустить программу, убедиться, что выставлены нужные параметры измерения, и послать команду на соединение. Установка соединения произойдет в соответствии с заданными параметрами, в ответ аппарат должен сообщить о статусе соединения.

Для корректного измерения нужно выполнить калибровку ВАЦ. Запуск калибровки осуществляется при помощи измерения мер калибровки. В процессе калибровки требуется подсоединить их по очереди сначала к одному порту, затем к другому. После калибровки коэффициента отражения, требуется откалибровать коэффициент передачи соединив кабелем порты между собой. Полученные данные автоматически применяются к измеряемым данным.

Для измерения нужно отправить команду на измерение. Полученные данные будут скорректированы при помощи калибровки.

2.2 Функции используемые в ПО

Для корректной работы с ВАЦ необходимо написать функции помогающие использовать библиотеку совместно с графическим интерфейсом.

2.2.1 Функция для соединения с устройством

Работа с ВАЦ возможна при корректной инициализации. Функция соединения позволяет установить связь с ним. Для соединения отправляется команда. При успешном соединении произойдет загрузка параметров калибровки и параметров измерения, к которым сделана калибровка. Параметры измерения и калибровки автоматически применяются к текущему измерению. Алгоритм работы функции соединения представлен на рисунке 7.

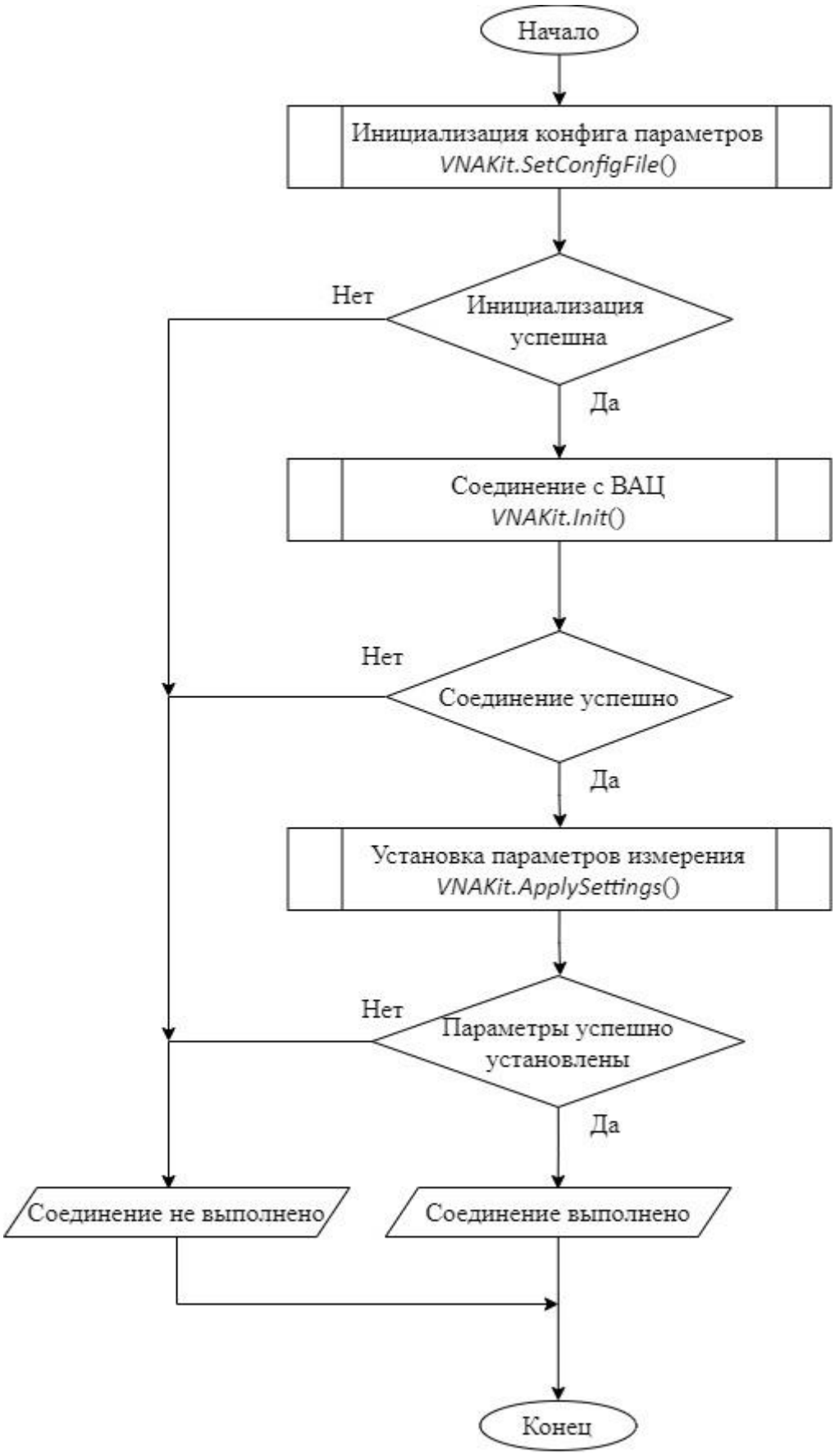


Рисунок 7 – Алгоритм работы функции соединения

Исходный код функции соединения представлен в приложении А.

2.2.2 Функция калибровки аппарата

Метод калибровки – это процедура, в которой измеряется набор стандартизованных компонентов (калибровочный набор) и используется для построения модели исправления ошибок. Набор калибровочных стандартов включают в себя короткие, открытые, нагружающие и сквозные компоненты.

Компоненты *short*, *open* и *load* являются однопортовыми устройствами, которые канонически имеют коэффициенты отражения Г*short* = −1, Г*open* = 1, Г*load* = 0. Эти три компонента измеряются для решения трехчленной однопортовой модели при векторной коррекции ошибок.

Идеальным прямым соединением является линия передачи нулевой длины без отражения и с полной передачей. Но настоящее прямое соединение будет иметь конечную длину, рассогласование импеданса и потери при передаче. Эта мера используется для получения большинства параметров передачи для моделей ошибок. Другие группы стандартов (которые используются в других методах калибровки) предпочитают использовать линию передачи фиксированной длины, называемую «линейным» стандартом.

Метод *SOLT* (*Short-Open-Load-Thru*) требует измерения стандартов *SOL* (*Short-Open-Load*) на каждом порту вместе со стандартом прямого соединения между каждым портом. Если порты могут быть соединены вместе напрямую, прямое соединение не требуется.

Функция калибровки выполняет процедуру метода *SOLT*. При успешном соединении с аппаратом возможно провести двухпортовую калибровку *SOLT* с заданными параметрами измерения. Пользователю предлагается провести семь измерений, чтобы построить 12-членную модель коррекции. Пример процедуры метода *SOLT* приведен на рисунке 8.

```
>> Measure OPEN @ Port-1:
Recording...Done.

>> Measure SHORT @ Port-1:
Recording...Done.

>> Measure LOAD @ Port-1:
Recording...Done.

-- Switch to Port-2 --

>> Measure OPEN @ Port-2:
Recording...Done.

>> Measure SHORT @ Port-2:
Recording...Done.

>> Measure LOAD @ Port-2:
Recording...Done.

>> Measure THRU:
Recording...Done.

Constructing 12-Term Error Model...
```

Рисунок 8 – Процедура двухпортовой калибровки *SOLT*

Алгоритм работы функции калибровки изображен на рисунке 9.

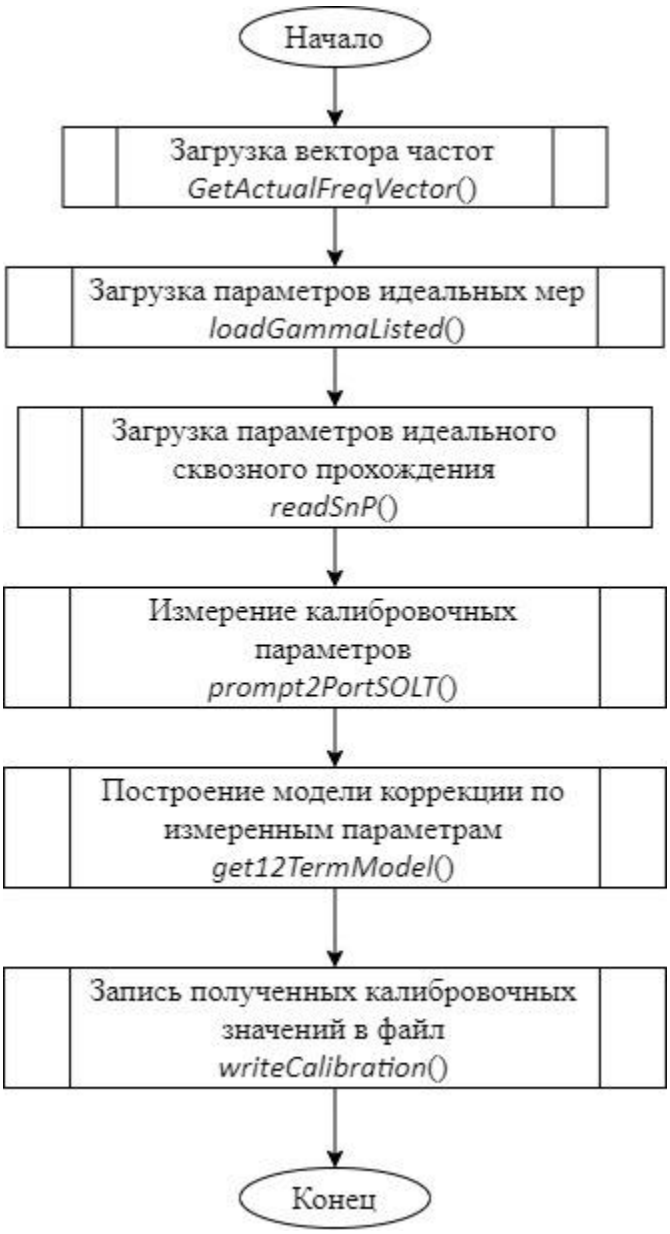


Рисунок 9 – Алгоритм работы функции калибровки

Исходный код функции калибровки представлен в приложении Б.

2.2.3 Функция измерения

Функция измерения производит считывание данных с устройства в соответствии с заданными параметрами измерения и преобразует их в *S*-параметры с помощью встроенной библиотеки. Полученные данные корректируются при помощи данных коррекций, полученных ранее при калибровке. Если калибровка не была произведена в окне состояния отобразится надпись «Нужна калибровка». Далее данные преобразуются при помощи формул в нужные параметры в зависимости от выбранного способа отображения. Для

корректного отображения маркеров производится интерполяция данных. При помощи встроенной функции «*plot*», происходит построение графиков. После вывода данных на экран происходит обновление данных маркеров если они были установлены.

Для работы с прибором в режиме онлайн функция вызывается с помощью таймера.

2.2.4 Функция расчета параметров маркера

Для удобного использования данных представлена возможность добавления маркеров (до пяти штук) и определения полосы по заданному уровню.

Добавление маркера вызывает функцию расчета параметра маркеров. Для определения более точного уровня маркера относительно полосы, производится интерполяция данных уровня и частоты. Относительно заданной частоты рассчитывается уровень маркера. В зависимости от выбранного числа маркеров. Происходит их отображение на графике и на экране графического интерфейса.

Расчет полосы по уровню происходит аналогичным способом за исключением расчета уровня маркера. Расчет полосы осуществляется за счет поиска крайних точек полосы, при которых график опускается ниже заданного уровня. После расчета значение полосы выводится на экран графического интерфейса, а на графике маркерами отображаются крайние точки полосы.

2.3 Интерфейс ПО

Графический интерфейс ВАЦ написан при помощи программного продукта «*App Designer*» компании «*MathWorks*».

App Designer позволяет создавать профессиональные приложения. Перетаскивая визуальные компоненты, можно разметить дизайн графического пользовательского интерфейса (*GUI*), используя встроенный редактор, можно быстро запрограммировать его поведение.

Вид разработанного графического интерфейса изображен на рисунке 10.

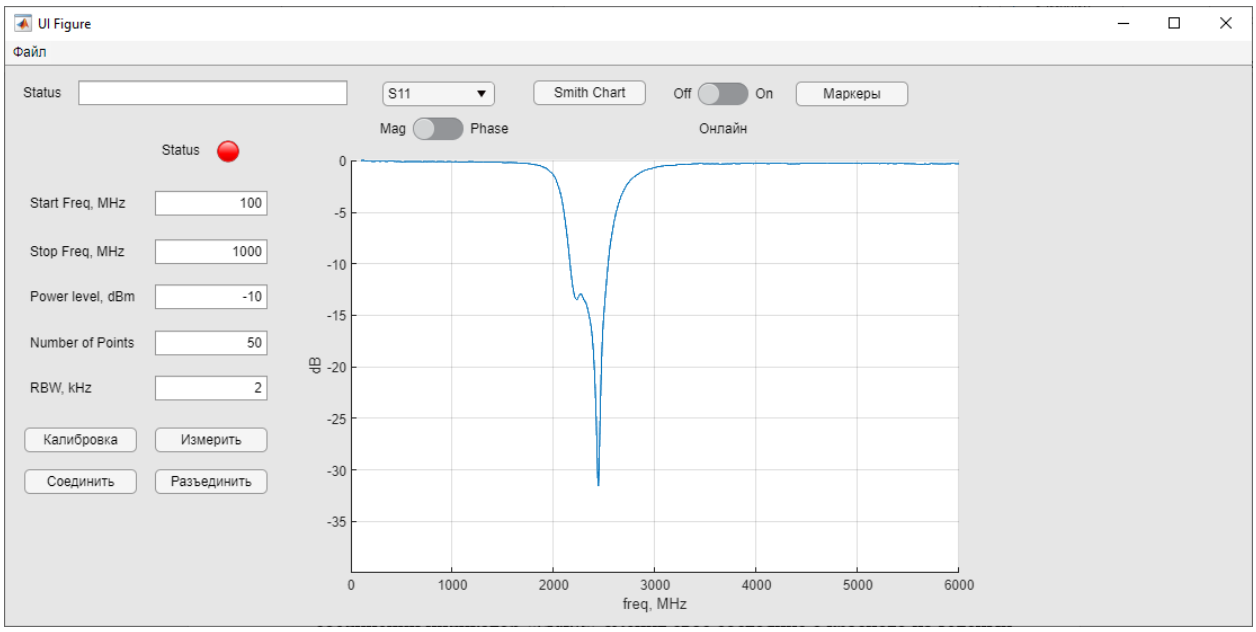


Рисунок 10 – Вид графического интерфейса

Для соединения нужно нажать на кнопку «Соединить», при корректном соединении индикатор «*Status*» сменит свое состояние с красного на зеленый. Кнопка «Соединить» и статус соединения показаны на рисунке 10.

Начальная частота измерения задается в поле «*Start Freq, MHz*» в диапазоне от 100 до 6000 МГц. Конечная частота измерения задается в поле «*Stop Freq, MHz*» в диапазоне от 100 до 6000 МГц. Уровень выходной мощности генератора задается в поле «*Power level, dBm*» в диапазоне от минус 10 до 0 дБм. Количество точек измерения задается в поле «*Number of Points*» в диапазоне от 2 до 1001. Полоса пропускания приемника задается в поле «*RBW, kHz*» в диапазоне от 2 до 140 кГц. Поля параметров измерения изображены на рисунке 11.

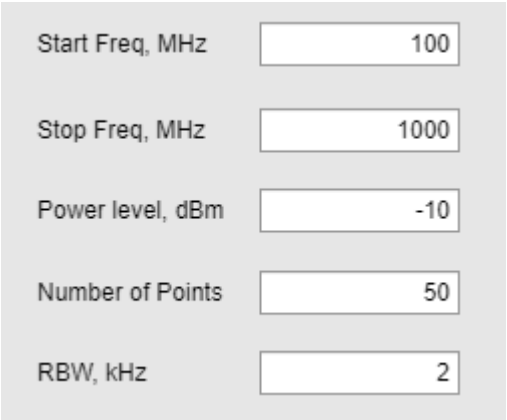


Рисунок 11 – Поля параметров измерения

Кнопка «Калибровка», изображенная внизу окна на рисунке 10, позволяет запустить процедуру метода *SOLT* на заданных параметрах измерения. После нажатия на кнопку программа перейдет в консольное окно программы *Matlab* и начнется калибровка. Пользователю предстоит провести семь измерений. Пример калибровки изображен на рисунке 8.

Кнопка «Разъединить» позволяет сбросить параметры инициализации. Индикатор «*Status*» сменит свое состояние с зеленого на красный.

Для открытия файлов нужно вверху окна перейти в меню «Файл» и нажать кнопку «Открыть». После нажатия отобразится диалоговое окно, в котором нужно выбрать файл. Как только файл будет открыт он сразу отобразится на графике, с заданными параметрами. Пример открытия файла показан на рисунке 12.

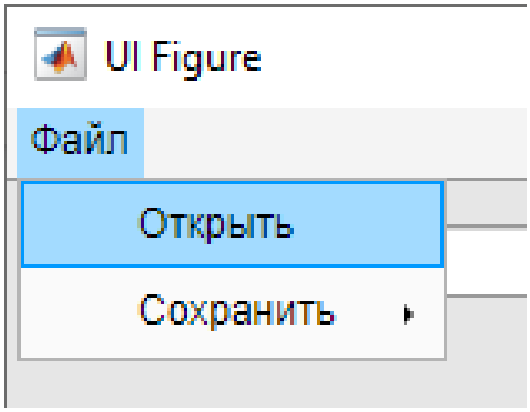


Рисунок 12 – Пример открытия графика

Для сохранения графика нужно вверху окна перейти в меню «Файл > Сохранить» и выбрать нужный из форматов для сохранения измеренных данных. Сохранить измеренные данные можно в двух форматах. Первым форматом является «*Touchstone file*». Данный формат является наиболее популярным для обмена информацией между системами компьютерного моделирования и измерительной аппаратурой. Данный формат обозначается как *SnP*, где *n* – это количество портов измерения. В нашем случае будет выводиться формат «**.S2P*».

Вторым форматом является формат «*CSV*». Данный формат является текстовым и служит для предоставления данных в табличном виде. Данный формат будет выводиться «*.*csv*».

После выбора формата файл с данными сохранится в корневом каталоге в папке «*measure*» с именем «*dd mmm yyyy hh-mm-ss -measure*» с расширением «.*S2P*» или «.*csv*». Пример сохранения данных изображен на рисунке 13.

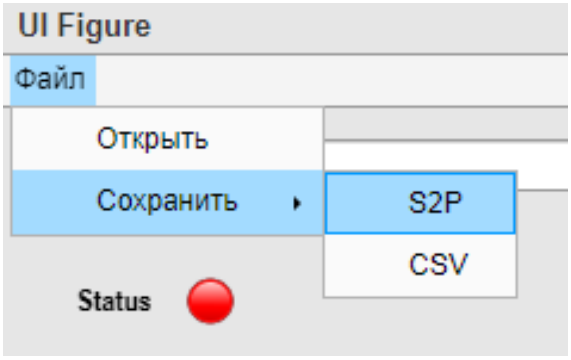


Рисунок 13 – Пример сохранения графика

Кнопка «Измерить» позволяет сделать единичное измерение. После нажатия на кнопку будет выполнено измерение параметров присоединенного к ВАЦ устройства. Полученные данные будут откалиброваны с помощью корректий, полученных при калибровке. Если калибровка не осуществлялась аппарат в окне статуса отобразит надпись «Нужна калибровка». Откалиброванные значения будут отображены на графике в соответствии с выбранным режимом отображения данных, изображенном на рисунке 14.

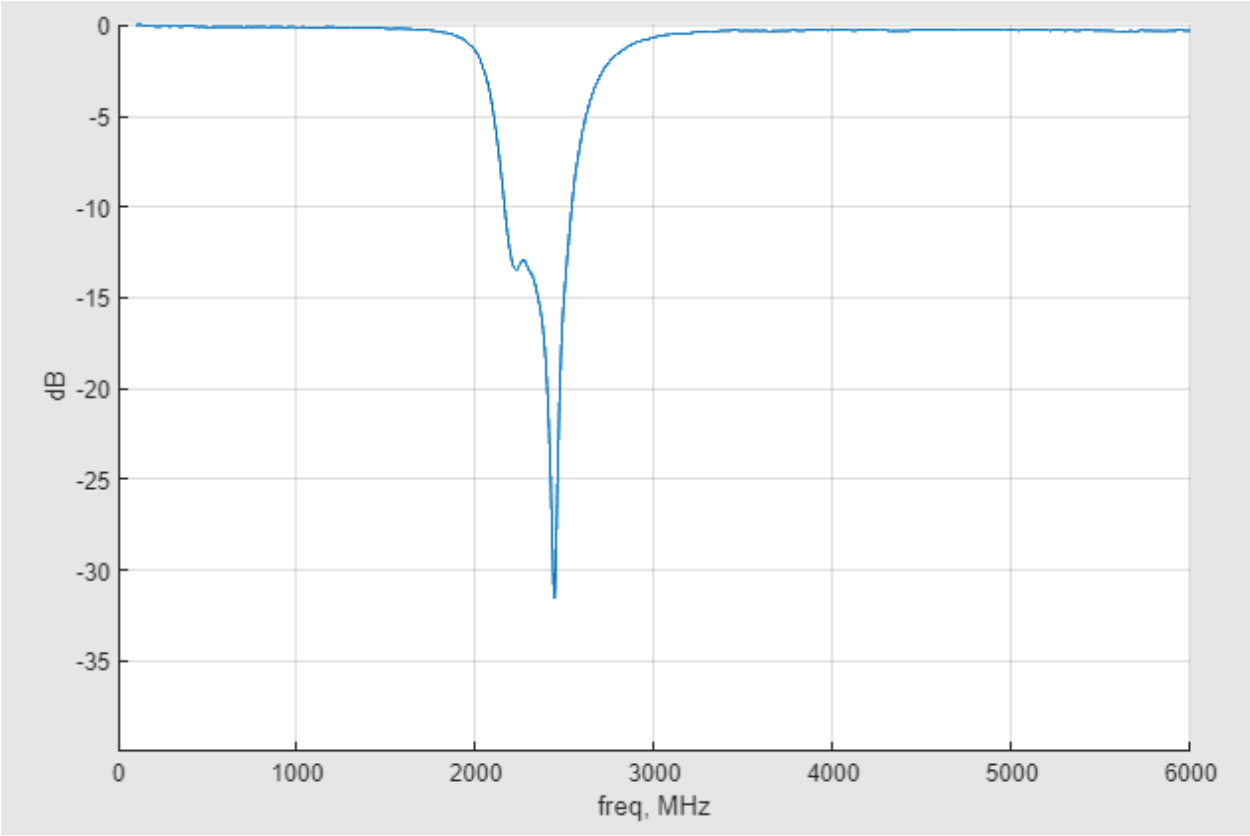


Рисунок 14 – График измеренных характеристик КСВ в зависимости от частоты

После отображения графика будут обновлены маркеры, если они были установлены.

Режим отображения данных выбирается в выпадающем списке. Доступны такие режимы отображения как:

– коэффициент отражения от первого входа (S_{11}), измеряемый в децибелах;

– коэффициент отражения от первого входа (S_{22}), измеряемый в децибелах;

– коэффициент передачи с первого порта на второй (S_{21}), измеряемый в децибелах;

– коэффициент передачи с второго порта на первый (S_{12}), измеряемый в децибелах;

– коэффициент стоячей волны (*VSWR*) для обеих портов;

– график изменения фазы на первом порте;

– график изменения фазы на втором порте;

– график изменения фазы при переходе сигнала с одного порта на другой;

– отображение реальной и мнимой части измеренных данных на диаграмме Вольперта-Смита.

Выбор режима отображения измеренных данных показан на рисунке 15.

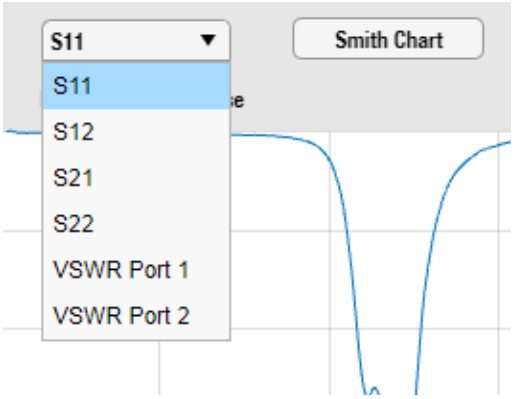


Рисунок 15 – Выбор режима отображения параметров

Выбор между магнитудой и фазой изображен на рисунке 16.

Рисунок 16 – Переключатель вывода на график фазы или магнитуды

При помощи переключателя «Онлайн» можно включить измерение в режиме онлайн, при условии, что соединение аппарата с компьютером установлено. Режим подразумевает выполнение единичного измерения периодично через фиксированные промежутки времени. При запущенном режиме невозможно изменить параметры измерения. Чтобы изменить параметры измерения нужно выключить режим измерения онлайн. Переключатель режима измерения изображен на рисунке 17.

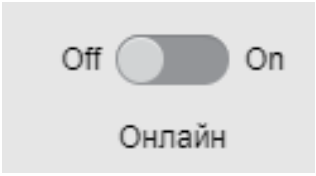


Рисунок 17 – Переключатель режима измерения

Для установки маркера нужно нажать кнопку «маркеры». В открывшемся диалоговом окне выбрать частоту и нажать добавить маркер. При добавлении маркера на графике высветится маркер на выбранной частоте и справа от графика отобразится частота и соответствующий ей уровень. Возможно добавление до пяти маркеров на график. Пример добавления и отображения маркера показан на рисунке 18.

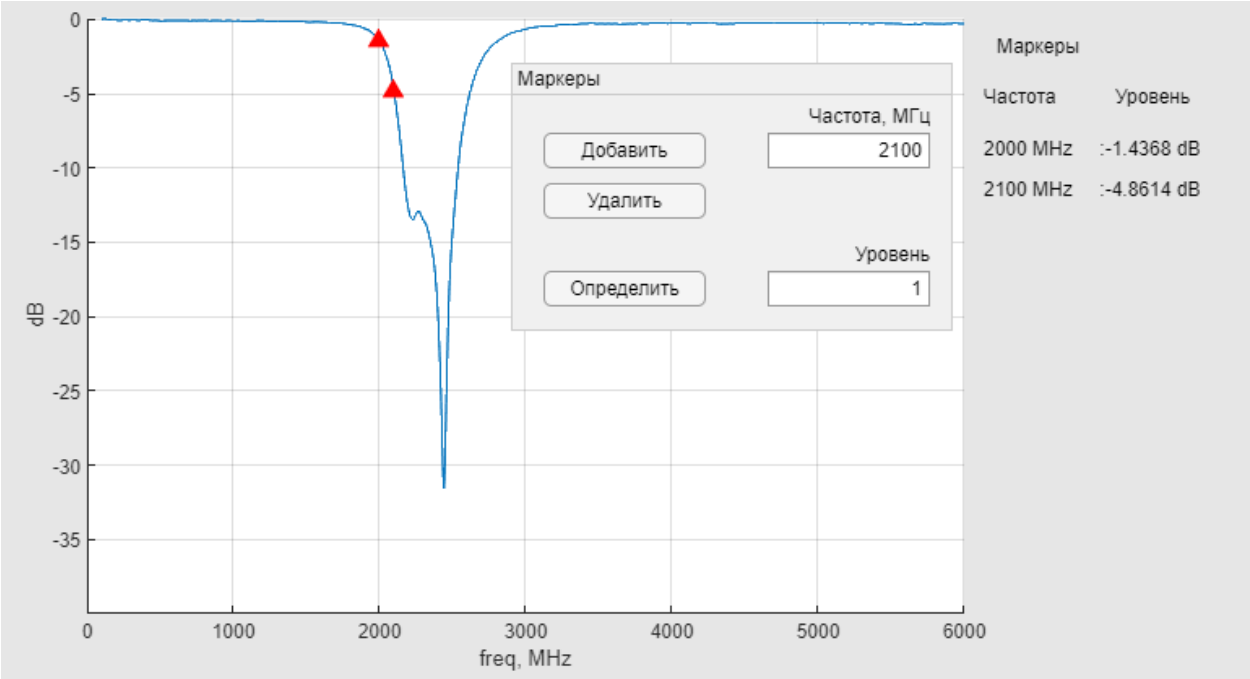


Рисунок 18 – Пример добавления маркера на график

Для измерения полосы частот по уровню нужно в диалоговом окне «Маркеры» ввести уровень и нажать кнопку «Определить». Пример измерения полосы частот по уровню показан на рисунке 19.

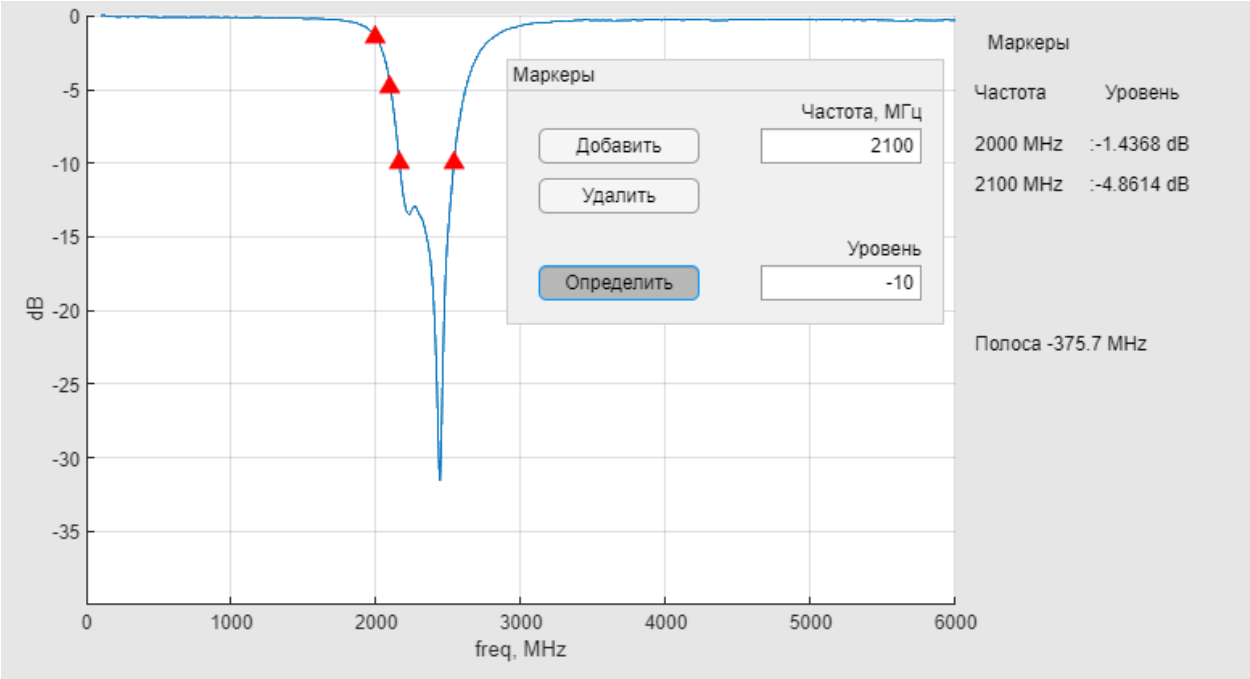


Рисунок 19 – Пример добавления маркера на график

После нажатия на кнопку справа от графика высветится измеренная полоса. На графике добавятся маркеры по краям полосы частот. Алгоритм работы графического интерфейса изображена на рисунке 20.

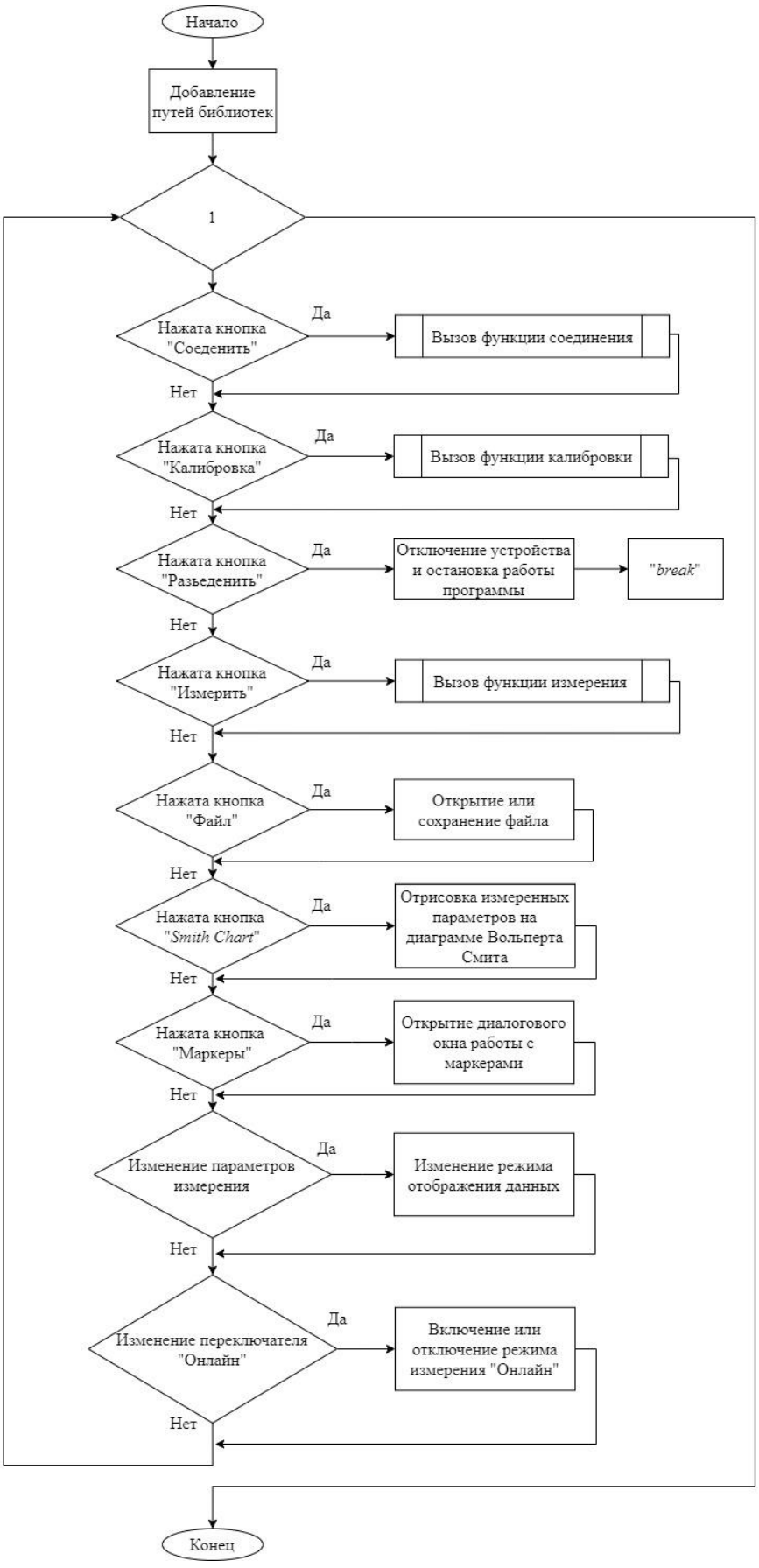


Рисунок 20 – Алгоритм работы графического интерфейса

Код графического приложения представлен в приложении В.

3 Доработка методических указаний

При помощи разработанного программного обеспечения были доработаны методические указания для лабораторных работ по дисциплине «Устройства СВЧ и антенны» [5].

В состав лабораторной установки входит обучающий комплект ВАЦ *UVNA*-63, набор калибровочных мер, компьютер с установленной операционной системой *Windows x*64 и с предустановленной программой *MATLAB*. Для работы разработанного ПО важно установить официальное программное обеспечение «*VNAKit.exe*» [6].

Для доработки были выбраны лабораторные работы №3, 4, 8 из [5]. В данных лабораторных работах исследуются устройства на частотах ниже 6 ГГц, что входит в диапазон измерения аппарата. Доработанные методические указания упростят выполнение лабораторных работ. Возможность измерения в полосе частот от 100 до 6000 МГц и измерение как в однопортовом так и в двухпортовом режиме позволит выполнять все лабораторные работы на одном устройстве. Добавление маркеров и определение полосы частот по уровню упростят измерение параметров исследуемого устройства. Доработанные методические указания приведены в Приложении Г.

ЗАКЛЮЧЕНИЕ

Разработанный графический интерфейс и разработанные функции измерения имеют открытый исходный код, поэтому возможно дополнение функциями и развитие проекта в дальнейшем.

Разработанное программное обеспечение рекомендуется использовать в обучающих целях при выполнении лабораторных работ и измерениях исследуемых устройств. Поставленные задачи решены полностью.

Рекомендуется для увеличения скорости измерения модернизировать или разработать новый драйвер устройства. При увеличении скорости измерения возможна более плавная работа интерфейса.

Исследуемое устройство является учебным комплектом, и не имеет в своем комплекте программу позволяющую отображать измеренные данные на графике, поэтому разработанное программное обеспечение решает вопрос использования устройства сразу при покупке. Аналогов программного обеспечения не существует.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ВЕКТОРНОГО АНАЛИЗА ЦЕПЕЙ В СООТВЕТСТВИИ С МИ 3411-2013 [Электронный ресурс] – *URL*: *http://www.planarchel.ru/Products/Measurement%20instrument/paper_2014-07 18_2.pdf* (дата обращения: 20.01.2020).

2. *UVNA-63 University Vector Network Analyzer Kit* [Электронный ресурс]. – *URL*: *https://www.minicircuits.com/WebStore/dashboard.html? model=UVNA-63* (дата обращения: 20.02.2021).

3. *UVNA*-63 *Application Note Error Correction* [Электронный ресурс]. – *URL*: *https://www.minicircuits.com/app/AN49-016.pdf* (дата обращения: 20.03.2021).

4. *Software and programming guide* [Электронный ресурс]. – *URL*: *https://www.minicircuits.com/pdfs/UVNA-63_Software_and_Program-ming_Guide.pdf* (дата обращения: 20.03.2021).

5. Расчет и измерение характеристик устройств СВЧ и антенн : учебное пособие / Ю. Е. Мительман, Р. Р. Абдуллин, С. Г. Сычугов, С. Н. Шабунин; под общ. ред. канд. техн. наук, доц. Ю. Е. Мительмана. – Екатеринбург : Изд-во Урал. ун-та, 2016. – 140 с.

6. *VnaKitSetup* [Электронный ресурс]. – *URL*: *https://www.minicircuits.com/softwaredownload/VnaKitSetup_1.0.11.zip* (дата обращения: 20.03.2021).

ПРИЛОЖЕНИЕ А

Код функции соединения

```
        result = board.SetConfigFile('C:/Program Files/Vayyar/VNAKit/bin/.config');
        if(result ~= VNAKit.VNAKIT_RESULT.VNAKIT_SUCCESS)
            return;
        end
        result = board.Init();
        if(result ~= VNAKit.VNAKIT_RESULT.VNAKIT_SUCCESS)
            return;
        end

        [LF,UF,PTS,RBW,PWR,TXTR,MODE]                                                    =
deal(fstart,fstop,nump,rbw,power,txtr,mode);
        settings = recordingSettings(LF,UF,PTS,RBW,PWR,TXTR,MODE);

        result = board.ApplySettings(settings);

        fprintf('The board is initialized with settings:\n');
        fprintf(getSettingsStr(settings));
```

ПРИЛОЖЕНИЕ Б

Код функции калибровки

```
function status = CalibrationBoard(board, settings)

ports = portMap();
% Returns the frequency vector in MHz rounded
% to the nearest allowable frequency point
freq_vec = double(board.GetActualFreqVector());
freq_vec_Hz = 1e6*freq_vec;

% [LF,UF,PTS,RBW,PWR] = deal(100,6000,236,2,-10);
% settings = recordingSettings(LF,UF,PTS,RBW,PWR);

% These are the file paths and file names for the Data based calibration
refl_file = {'lib\stds\open.S1P','lib\stds\short.S1P','lib\stds\load.S1P'};
thru_file = 'lib\stds\thru.S2P';

% Load the S-parameters from the data files and perform a linear
% interpolation of the data for use in the error models
[Gamma_listed] = loadGammaListed(refl_file,freq_vec_Hz);
[T,~] = readSnP(thru_file,freq_vec_Hz);

% Prompt the user through a two port Calibration

[Gm1,Gm2,Tm,~] = prompt2PortSOLT(board,settings,ports);

% Construct the 12-term model from the standards data and the data
% measured from the prompt

[fwd_terms,rev_terms] =
get12TermModel(Gamma_listed,Gm1,Gamma_listed,Gm2,T,Tm);

start = settings.freqRange.freqStartMHz;
stop = settings.freqRange.freqStopMHz;
num = settings.freqRange.numFreqPoints;
rbw = settings.rbw_khz;
powerlvl = settings.outputPower_dbm;

writeCalibration(fwd_terms, rev_terms,start,stop,num,rbw,powerlvl);

status = true;

end
```

ПРИЛОЖЕНИЕ В

Код графического интерфейса

```
classdef app1 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure              matlab.ui.Figure
        Button_measure        matlab.ui.control.Button
        Button_calibr         matlab.ui.control.Button
        StartFreqMHzLabel     matlab.ui.control.Label
        StopFreqMHzLabel      matlab.ui.control.Label
        PowerleveldBmLabel    matlab.ui.control.Label
        NumberofPointsLabel   matlab.ui.control.Label
        RBWkHzLabel           matlab.ui.control.Label
        StatusLampLabel       matlab.ui.control.Label
        StatusLamp            matlab.ui.control.Lamp
        Button_connect        matlab.ui.control.Button
        StatusEditFieldLabel  matlab.ui.control.Label
        StatusEditField       matlab.ui.control.EditField
        Button_disconnect     matlab.ui.control.Button
        DropDown              matlab.ui.control.DropDown
        EditField_StartFreq   matlab.ui.control.NumericEditField
        EditField_StopFreq    matlab.ui.control.NumericEditField
        EditField_PowerLvl    matlab.ui.control.NumericEditField
        EditField_NumPoint    matlab.ui.control.NumericEditField
        EditField_RBW         matlab.ui.control.NumericEditField
        OnlineMeasSwitchLabel matlab.ui.control.Label
        OnlineMeasSwitch      matlab.ui.control.Switch
        Button_Save           matlab.ui.control.Button
        Button_Open           matlab.ui.control.Button
        S2PLabel              matlab.ui.control.Label
        SmithChartButton_2    matlab.ui.control.StateButton
        Button_markers        matlab.ui.control.Button
        Panel                 matlab.ui.container.Panel
        Button_add            matlab.ui.control.Button
        Button_del            matlab.ui.control.Button
        Label                 matlab.ui.control.Label
        EditField_freq        matlab.ui.control.NumericEditField
        Button_band           matlab.ui.control.StateButton
        Label_2               matlab.ui.control.Label
        EditField_lvl         matlab.ui.control.NumericEditField
        Label_marker          matlab.ui.control.Label
        Label_freq            matlab.ui.control.Label
        Label_level           matlab.ui.control.Label
        Label_marker_1        matlab.ui.control.Label
        Label_marker_2        matlab.ui.control.Label
```

```
            Label_marker_3        matlab.ui.control.Label
            Label_marker_4        matlab.ui.control.Label
            Label_marker_5        matlab.ui.control.Label
            Label_marker_6        matlab.ui.control.Label
            Switch              matlab.ui.control.Switch
            UIAxes               matlab.ui.control.UIAxes
        end


        properties (Access = private)
            WindSpeedTimer % Timer object
        end

        methods (Access = private)
    %
=================================================================
            function markerset(app)
                global markercnt;
                global freqmarker;
                global levelmarker;

                global S_param_meas;
                global freq;
                global plot_p;

                flagSmith = app.SmithChartButton_2.Value;


                plot_p.Marker = 'none';


                B_band = app.Button_band.Value;
                if(markercnt > 0) || (B_band == true)
                    freqdel = 100000;                           % частота дискретов измерений
                    newfreq = [freq(1):freqdel:freq(end)];
                    Measur = app.DropDown.Value;

                    switch Measur
                        case "S11"
                            nameexp = ' dB';
                            Spar = S_param_meas(1,1,:);
                            Spar = Spar(:,:);
                            new_spar = interp1(freq,Spar,newfreq,'spline');
                            param = 20*log10(abs(new_spar));
                        case "S12"
                            nameexp = ' dB';
                            Spar = S_param_meas(1,2,:);
                            Spar = Spar(:,:);
                            new_spar = interp1(freq,Spar,newfreq,'spline');
                            param = 20*log10(abs(new_spar));
                        case "S21"
                            nameexp = ' dB';
```

```
            Spar = S_param_meas(2,1,:);
            Spar = Spar(:,:);
            new_spar = interp1(freq,Spar,newfreq,'spline');
            param = 20*log10(abs(new_spar));
        case "S22"
            nameexp = ' dB';
            Spar = S_param_meas(2,2,:);
            Spar = Spar(:,:);
            new_spar = interp1(freq,Spar,newfreq,'spline');
            param = 20*log10(abs(new_spar));
        case "VSWR Port 1"
            nameexp = '';
            Spar = S_param_meas(1,1,:);
            Spar = Spar(:,:);
            new_spar = interp1(freq,Spar,newfreq,'spline');
            Sdb=20*log10(abs(new_spar));
            G=10.^(Sdb/20);
            param=(1+G)./(1-G);
        case "VSWR Port 2"
            nameexp = '';
            Spar = S_param_meas(2,2,:);
            Spar = Spar(:,:);
            new_spar = interp1(freq,Spar,newfreq,'spline');
            Sdb=20*log10(abs(new_spar));
            G=10.^(Sdb/20);
            param=(1+G)./(1-G);
    end

    if (flagSmith == true)
        param = new_spar;
        nameexp = '';
    end

    if(strcmp(app.Switch.Value,'Mag') == false)
        param = 180*(angle(new_spar))/pi;
        nameexp = ' deg';
    end

    for i = 1:1:markercnt
        a = (((freqmarker(i)*10^6) - newfreq(1))/freqdel) + 1;     % номер измерения
        levelmarker(i) =param(a);
        name(i) = strcat(string(freqmarker(i)),' MHz :',string(lev-
elmarker(i)),nameexp); %  название маркера для отображения
        mark(i) = a;
    end

    maxmin = 0;
    lvl = app.EditField_lvl.Value;

    startfreq = 1;
    stopfreq = 1;
```

```
for i = 1:1:size(param,2)
   if(maxmin == 0)
      if(param(i) <= lvl)
         startfreq = i;
         maxmin = 1;
      end
   else
      if(param(i) >= lvl)
         stopfreq = i;
         break;
      end
   end

end


if(markercnt == 1)
   app.Label_marker_1.Text = name(1); % string(levelmarker(1))];
   app.Label_marker_2.Text = '';
   app.Label_marker_3.Text = '';
   app.Label_marker_4.Text = '';
   app.Label_marker_5.Text = '';
end
if(markercnt == 2)
   app.Label_marker_1.Text = name(1);
   app.Label_marker_2.Text = name(2);
   app.Label_marker_3.Text = '';
   app.Label_marker_4.Text = '';
   app.Label_marker_5.Text = '';
end
if(markercnt == 3)
   app.Label_marker_1.Text = name(1);
   app.Label_marker_2.Text = name(2);
   app.Label_marker_3.Text = name(3);
   app.Label_marker_4.Text = '';
   app.Label_marker_5.Text = '';
end
if(markercnt == 4)
   app.Label_marker_1.Text = name(1);
   app.Label_marker_2.Text = name(2);
   app.Label_marker_3.Text = name(3);
   app.Label_marker_4.Text = name(4);
   app.Label_marker_5.Text = '';
end
if(markercnt == 5)
   app.Label_marker_1.Text = name(1);
   app.Label_marker_2.Text = name(2);
   app.Label_marker_3.Text = name(3);
   app.Label_marker_4.Text = name(4);
   app.Label_marker_5.Text = name(5);
end
```

```
%               plot_p.MarkerSize = 5;
%               plot_p.Marker = '*';
%               plot_p.MarkerEdgeColor = 'red';
%               plot_p.MarkerFaceColor = 'red';
%               plot_p.MarkerIndices = mark;

                if(B_band == true)
                   band = (newfreq(stopfreq) - newfreq(startfreq))/10^6;
                   app.Label_marker_6.Text = strcat("Полоса -",string(band), " MHz");
                   mark(markercnt+1) = startfreq;
                   mark(markercnt+2) = stopfreq;
                else
                   app.Label_marker_6.Text = '';
                end

                if (flagSmith == true)
%                     plot_p.MarkerSize = 8;
%                     plot_p.Marker = '^';
%                     plot_p.Marker = mark;
                else
                   plot_p.MarkerSize = 8;
                   plot_p.Marker = '^';
                   plot_p.MarkerEdgeColor = 'red';
                   plot_p.MarkerFaceColor = 'red';
                   plot_p.MarkerIndices = mark;
                end


            end

        end
%
==========================================================================
        function param = read_sparam(app)
            global Board;
            global SettingsBoard;
            global ports;
            global freq;
            global flag_calib;
            global fwd_terms;
            global rev_terms;

            if(flag_calib == 0)
               app.StatusEditField.Value = "Нужна калибровка" ;
               pause(0.1);
            end


            [rec_tx1,rec_tx2] = measure2Port(Board,SettingsBoard,ports);
            S_param_meas = ab2S(rec_tx1,rec_tx2,ports);

            if(flag_calib == 1)
```

```
            S_param = correct12Term(S_param_meas,fwd_terms,rev_terms);
            param = S_param;
        else
            param = S_param_meas;
        end

        freq_vec = double(Board.GetActualFreqVector());
        freq = 1e6*freq_vec;



    end
%
==========================================================================
    function measure_param(app, S_param)
%       global Measur;
        global flagSmith;
        global freq;
        global plot_p;

        flagSmith = app.SmithChartButton_2.Value;

        Measur = app.DropDown.Value;


        switch Measur
            case "S11"
                param = [1, 1];
            case "S12"
                param = [1, 2];
            case "S21"
                param = [2, 1];
            case "S22"
                param = [2, 2];
            case "VSWR Port 1"
                param = [1, 1];
                app.SmithChartButton_2.Value = false;
                flagSmith = false;
            case "VSWR Port 2"
                param = [2, 2];
                app.SmithChartButton_2.Value = false;
                flagSmith = false;
        end


%       numpoint = app.EditField_NumPoint.Value;

%       for i = 1:1:nump
%           Spar(i) = S_param(param(1),param(2),i);
%       end
%
        Spar = S_param(param(1),param(2),:);
```

```
            Spar = Spar(:,:);

            freqdel = 100000;                           % частота дискретов измерений
            newfreq = [freq(1):freqdel:freq(end)];
            new_spar = interp1(freq,Spar,newfreq,'spline');

        %       tfdataphase = 180*unwrap(angle(tfdata))/pi;     % unwrap- для разворачивании
фазы

            if (flagSmith == true)

               plot_p = smithplot(app.UIAxes);
               add(plot_p,freq,Spar);

            else

               Sdb=20*log10(abs(new_spar));
               label = '';

               if (Measur == "VSWR Port 1") || (Measur == "VSWR Port 2")

                  G=10.^(Sdb/20);
                  vswr=(1+G)./(1-G);
                  paramdata = vswr;
               else
                  if(strcmp(app.Switch.Value,'Mag') == true)
                     paramdata = Sdb;
                     label = 'dB';
                  else
                     paramdata = 180*(angle(new_spar))/pi;
                     label = 'deg';
                  end
               end

               plot_p = plot(app.UIAxes,newfreq,paramdata);
               app.UIAxes.XLabel.String = 'freq, Hz';
               app.UIAxes.YLabel.String = label;
               app.UIAxes.XGrid = 'on';
               app.UIAxes.YGrid = 'on';
            end

            markerset(app);

         end
        %
==========================================================================
         function ErrTimerFcn(app,~, ~)

            app.StatusEditField.Value = "Err timer";

            stop(app.WindSpeedTimer);
```

```
        end
    %
================================================================
        function WindSpeedTimerFcn(app, ~, ~)
            global S_param_meas;

            start = tic;

            app.StatusEditField.Value = "start meas" ;
            pause(0.1);

            S_param_meas = read_sparam(app);

            measure_param(app, S_param_meas);


            if(strcmp(app.OnlineMeasSwitch.Value, 'Off'))
               stop(app.WindSpeedTimer);
            end

            stop1 = toc(start);
            app.StatusEditField.Value = string(stop1) + " sec end meas";

        end

    end


    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)

            addpath('lib','calib\','measure\');
            addpath('lib\hidden\','lib\stds\','lib\utils\');
    %           global Measur;
    %           Measur = "S11";
            global markercnt;
            global flag_calib;

            markercnt = 0;
            flag_calib = 0;
            % Create timer object
            app.WindSpeedTimer = timer(...
            'ExecutionMode', 'fixedRate', ...      % Run timer repeatedly
            'Period', 24, ...                      % Period is 5 seconds
            'BusyMode', 'drop',...                 % Queue timer callbacks when busy
            'TimerFcn', @app.WindSpeedTimerFcn, ...
            "ErrorFcn", @app.ErrTimerFcn);   % Callback that runs every period

        end
```

```
            % Callback function: Button_measure, EditField_RBW
            function Button_measurePushed(app, event)
                global S_param_meas;

                start = tic;

                app.StatusEditField.Value = "start meas" ;
                pause(0.1);

                S_param_meas = read_sparam(app);

                measure_param(app, S_param_meas);

                stops = toc(start);
                app.StatusEditField.Value = string(stops) + " sec end meas";

            end

            % Button pushed function: Button_calibr
            function Button_calibrPushed(app, event)
                global Board;
                global SettingsBoard;
                global ports;
                global fwd_terms;
                global rev_terms;
                global flag_calib;

                if(app.StatusLamp.Color(2) == 1)

        %           StartFreq   = 100;
        %           StopFreq    = 6000;
        %           Numpoint    = 1000;
        %           RBW         = app.EditField_RBW.Value;
        %           Powerlvl    = app.EditField_PowerLvl.Value;
        %           Mode        = VNAKit.Mode.VNAKIT_MODE_TWO_PORTS;
        %
        %           SettingsBoard = recordingSettings(StartFreq,StopFreq,Numpoint,RBW,Pow-
erlvl,ports('Tx1'),Mode);
        %           Board.ApplySettings(SettingsBoard);
        %
                    valueSwitch = string(app.OnlineMeasSwitch.Value);
                    if(valueSwitch == "Off")
                        app.StatusEditField.Value = "Calibr start";

                        status = CalibrationBoard(Board,SettingsBoard);

                        [fwd_terms,rev_terms,start,stop,num,rbw,powerlvl] = readCalibration();

                        app.EditField_StartFreq.Value   = start;
                        app.EditField_StopFreq.Value    = stop;
                        app.EditField_NumPoint.Value    = num;
```

```
                app.EditField_RBW.Value         = rbw;
                app.EditField_PowerLvl.Value    = powerlvl;


                if(status == true)
                  app.StatusEditField.Value = "Calibr good";
                  flag_calib = 1;
                else
                  app.StatusEditField.Value = "Calibr not";
                end

              end

%               StartFreq   = app.EditField_StartFreq.Value;
%               StopFreq    = app.EditField_StopFreq.Value;
%               Numpoint    = app.EditField_NumPoint.Value;
%               RBW         = app.EditField_RBW.Value;
%               Powerlvl    = app.EditField_PowerLvl.Value;
%               Mode        = VNAKit.Mode.VNAKIT_MODE_TWO_PORTS;
%
%               SettingsBoard = recordingSettings(StartFreq,StopFreq,Numpoint,RBW,Pow-
erlvl,ports('Tx1'),Mode);
%               Board.ApplySettings(SettingsBoard);


            end

        end

        % Button pushed function: Button_connect
        function Button_connectPushed(app, event)

            global Board;
            global SettingsBoard;
            global fwd_terms;
            global rev_terms;
            global ports;
            global freq;
            global flag_calib;
%           global Measur;

            modulePath = 'C:/Program Files/Vayyar/VNAKit/bin/VNAKit.CSharp.dll';
            NET.addAssembly(modulePath);
            import VNAKit.*;

%           Measur = "S11";
            Board = VNAKit.VNAKit();
            result = initBoard(Board);

            if(result ~= VNAKit.VNAKIT_RESULT.VNAKIT_SUCCESS)
              app.StatusLamp.Color = [1 0 0];
            else
```

```
            app.StatusLamp.Color = [0 1 0];
        end

        app.StatusEditField.Value = string(result);

%           StartFreq   = app.EditField_StartFreq.Value;
%           StopFreq    = app.EditField_StopFreq.Value;
%           Numpoint    = app.EditField_NumPoint.Value;
%           RBW         = app.EditField_RBW.Value;
%           Powerlvl    = app.EditField_PowerLvl.Value;
%           Mode        = VNAKit.Mode.VNAKIT_MODE_TWO_PORTS;

        if(result == VNAKit.VNAKIT_RESULT.VNAKIT_SUCCESS)
            flag_calib = 1;

            [fwd_terms,rev_terms,start,stop,num,rbw,powerlvl] = readCalibration();

            app.EditField_StartFreq.Value   = start;
            app.EditField_StopFreq.Value    = stop;
            app.EditField_NumPoint.Value    = num;
            app.EditField_RBW.Value         = rbw;
            app.EditField_PowerLvl.Value    = powerlvl;
            Mode        = VNAKit.Mode.VNAKIT_MODE_TWO_PORTS;

            ports = portMap();
            SettingsBoard           =           ParameterSetup(Board,start,stop,num,rbw,power-
lvl,ports('Tx1'),Mode);
            freq_vec = double(Board.GetActualFreqVector());
            freq = 1e6*freq_vec;

        end

        app.EditField_freq.Limits = [app.EditField_StartFreq.Value app.EditField_Stop-
Freq.Value];

    end

    % Button pushed function: Button_disconnect
    function Button_disconnectPushed(app, event)
        global Board;
        Board.SetConfigFile('C:/Program Files/Vayyar/VNAKit/bin/.config');

        app.StatusLamp.Color = [1 0 0];
        app.StatusEditField.Value = "Board disconnect";
    end

    % Value changed function: DropDown
    function DropDownValueChanged(app, event)
%           global Measur;
        global S_param_meas;

        valueSwitch = string(app.OnlineMeasSwitch.Value);
```

```
            Measur = app.DropDown.Value;

            if (Measur == "VSWR Port 1") || (Measur == "VSWR Port 2")
                app.Switch.Value = 'Mag';
                app.Switch.Visible = 'off';
            else
                app.Switch.Visible = 'on';
            end

            if(valueSwitch == "Off")

                measure_param(app, S_param_meas);
            end

        end

        % Value changed function: EditField_NumPoint,
        % EditField_PowerLvl, EditField_StartFreq,
        % EditField_StopFreq
        function EditField_StartFreqValueChanged(app, event)
            global ports;
            global Board;
            global SettingsBoard;
            global flag_calib;

            if(app.StatusLamp.Color(2) == 1)

                StartFreq   = app.EditField_StartFreq.Value;
                StopFreq    = app.EditField_StopFreq.Value;
                Numpoint    = app.EditField_NumPoint.Value;
                RBW         = app.EditField_RBW.Value;
                Powerlvl    = app.EditField_PowerLvl.Value;
                Mode        = VNAKit.Mode.VNAKIT_MODE_TWO_PORTS;

                SettingsBoard = recordingSettings(StartFreq,StopFreq,Numpoint,RBW,Power-
lvl,ports('Tx1'),Mode);
                result = Board.ApplySettings(SettingsBoard);
                app.StatusEditField.Value = string(result);

                app.EditField_freq.Limits = [app.EditField_StartFreq.Value app.Edit-
Field_StopFreq.Value];
                flag_calib = 0;
            end

        end

        % Value changed function: OnlineMeasSwitch
        function OnlineMeasSwitchValueChanged(app, event)
            value = string(app.OnlineMeasSwitch.Value);

            if(value == "On")
                if(app.StatusLamp.Color(2) == 1)
```

```
            % If timer is not running, start it
            if strcmp(app.WindSpeedTimer.Running, 'off')
                % Start timer

                app.EditField_StartFreq.Editable = 'off';
                app.EditField_StopFreq.Editable = 'off';
                app.EditField_NumPoint.Editable = 'off';
                app.EditField_RBW.Editable = 'off';
                app.EditField_PowerLvl.Editable = 'off';

                start(app.WindSpeedTimer);
            end
        else
            app.OnlineMeasSwitch.Value = "Off";
        end
    elseif(value == "Off")
        % Stop timer
        stop(app.WindSpeedTimer);

        app.EditField_StartFreq.Editable = 'on';
        app.EditField_StopFreq.Editable = 'on';
        app.EditField_NumPoint.Editable = 'on';
        app.EditField_RBW.Editable = 'on';
        app.EditField_PowerLvl.Editable = 'on';

    end
end

% Close request function: UIFigure
function UIFigureCloseRequest(app, event)
    global Board;

    % Stop timer, then delete timer and figure
    stop(app.WindSpeedTimer);
    delete(app.WindSpeedTimer);
    if(app.StatusLamp.Color(2) == 1)
        Board.SetConfigFile('C:/Program Files/Vayyar/VNAKit/bin/.config');
    end

    delete(app);
end

% Button pushed function: Button_Save
function Button_SavePushed(app, event)
global S_param_meas;
global freq;
%     filepath = '\measure\';
tnow = datetime();
%     path = '\'+string(tnow)+'measure.S2P';

filepath = 'measure\';
namepath = 'measure.S2P';
```

```
        tnow.Format = 'dd MMM yyyy HH-mm-ss -';

        if(app.StatusLamp.Color(2) == 1)
%           freq_vec = double(Board.GetActualFreqVector());
%           freq_vec_Hz = 1e6*freq_vec;

          writeSnP(freq,S_param_meas, [filepath char(tnow) namepath]);
        end
        end

        % Button pushed function: Button_Open
        function Button_OpenPushed(app, event)
          global S_param_meas;
          global freq;
          value = string(app.OnlineMeasSwitch.Value);

          if(value == "Off")
            [file,path] = uigetfile('*.S2P');
            if(double(file))

              pyt = string(path)+string(file);
              data = nport(pyt);
              S_param_meas = data.NetworkData.Parameters;
              freq = data.NetworkData.Frequencies;
              freq = freq';
              measure_param(app,S_param_meas);

              app.EditField_freq.Limits = [(freq(1)/10^6) (freq(end)/10^6)];
            end
          end
        end

        % Value changed function: SmithChartButton_2
        function SmithChartButton_2ValueChanged(app, event)
          global S_param_meas;
          global flagSmith;

          flagSmith = app.SmithChartButton_2.Value;

          value = string(app.OnlineMeasSwitch.Value);

          if(value == "Off")
            measure_param(app,S_param_meas);
          end
        end

        % Button pushed function: Button_markers
        function Button_markersPushed(app, event)

          if(app.Panel.Visible == "off")
            if strcmp(app.WindSpeedTimer.Running, 'on')
              stop(app.WindSpeedTimer);
```

```
            app.OnlineMeasSwitch.Value = 'Off';
        end
        app.Panel.Visible = 'on';
    else
        app.Panel.Visible = 'off';
    end
end

% Button pushed function: Button_add
function Button_addPushed(app, event)
    global markercnt;
    global freqmarker;

    switch markercnt
        case 0
            markercnt = 1;
            freqmarker(markercnt) = app.EditField_freq.Value;

            app.Label_level.Visible = 'on';
            app.Label_freq.Visible = 'on';
            app.Label_marker.Visible = 'on';

        case 1
            markercnt = 2;
            freqmarker(markercnt) = app.EditField_freq.Value;
        case 2
            markercnt = 3;
            freqmarker(markercnt) = app.EditField_freq.Value;
        case 3
            markercnt = 4;
            freqmarker(markercnt) = app.EditField_freq.Value;
        case 4
            markercnt = 5;
            freqmarker(markercnt) = app.EditField_freq.Value;
        case 5
            markercnt = 5;
            freqmarker(markercnt) = app.EditField_freq.Value;
    end

    markerset(app);

end

% Button pushed function: Button_del
function Button_delPushed(app, event)
    global markercnt;

    if(markercnt >= 1)
        markercnt = markercnt - 1;
    end

    if(markercnt == 0)
```

```
            app.Label_level.Visible = 'off';
            app.Label_freq.Visible = 'off';
            app.Label_marker.Visible = 'off';

            app.Label_marker_1.Text = '';
            app.Label_marker_2.Text = '';
            app.Label_marker_3.Text = '';
            app.Label_marker_4.Text = '';
            app.Label_marker_5.Text = '';
        end

        markerset(app);
    end

    % Value changed function: Button_band
    function Button_bandValueChanged(app, event)
        value = app.Button_band.Value;

        if(value == false)
            app.Label_marker_6.Text = '';
        end

        markerset(app);
    end

    % Value changed function: Switch
    function SwitchValueChanged(app, event)
        value = app.Switch.Value;
        global S_param_meas;

        valueSwitch = string(app.OnlineMeasSwitch.Value);

        if(valueSwitch == "Off")

            measure_param(app, S_param_meas);
        end
    end
end

% Component initialization
methods (Access = private)

    % Create UIFigure and components
    function createComponents(app)

        % Create UIFigure and hide until all components are created
        app.UIFigure = uifigure('Visible', 'off');
        app.UIFigure.Color = [0.902 0.902 0.902];
        app.UIFigure.Position = [100 100 1107 497];
        app.UIFigure.Name = 'UI Figure';
```

```
            app.UIFigure.CloseRequestFcn = createCallbackFcn(app, @UIFigure-
CloseRequest, true);

            % Create Button_measure
            app.Button_measure = uibutton(app.UIFigure, 'push');
            app.Button_measure.ButtonPushedFcn = createCallbackFcn(app, @But-
ton_measurePushed, true);
            app.Button_measure.Position = [134 155 100 22];
            app.Button_measure.Text = 'Измерить';

            % Create Button_calibr
            app.Button_calibr = uibutton(app.UIFigure, 'push');
            app.Button_calibr.ButtonPushedFcn = createCallbackFcn(app, @Button_cali-
brPushed, true);
            app.Button_calibr.Position = [18 155 100 22];
            app.Button_calibr.Text = 'Калибровка';

            % Create StartFreqMHzLabel
            app.StartFreqMHzLabel = uilabel(app.UIFigure);
            app.StartFreqMHzLabel.Position = [23 365 90 22];
            app.StartFreqMHzLabel.Text = 'Start Freq, MHz';

            % Create StopFreqMHzLabel
            app.StopFreqMHzLabel = uilabel(app.UIFigure);
            app.StopFreqMHzLabel.Position = [23 322 90 22];
            app.StopFreqMHzLabel.Text = 'Stop Freq, MHz';

            % Create PowerleveldBmLabel
            app.PowerleveldBmLabel = uilabel(app.UIFigure);
            app.PowerleveldBmLabel.Position = [23 282 99 22];
            app.PowerleveldBmLabel.Text = 'Power level, dBm';

            % Create NumberofPointsLabel
            app.NumberofPointsLabel = uilabel(app.UIFigure);
            app.NumberofPointsLabel.Position = [23 241 98 22];
            app.NumberofPointsLabel.Text = 'Number of Points';

            % Create RBWkHzLabel
            app.RBWkHzLabel = uilabel(app.UIFigure);
            app.RBWkHzLabel.Position = [23 201 60 22];
            app.RBWkHzLabel.Text = 'RBW, kHz';

            % Create StatusLampLabel
            app.StatusLampLabel = uilabel(app.UIFigure);
            app.StatusLampLabel.HorizontalAlignment = 'right';
            app.StatusLampLabel.Position = [134 412 40 22];
            app.StatusLampLabel.Text = 'Status';

            % Create StatusLamp
            app.StatusLamp = uilamp(app.UIFigure);
            app.StatusLamp.Position = [189 412 20 20];
            app.StatusLamp.Color = [1 0 0];
```

```
            % Create Button_connect
            app.Button_connect = uibutton(app.UIFigure, 'push');
            app.Button_connect.ButtonPushedFcn = createCallbackFcn(app, @Button_con-
nectPushed, true);
            app.Button_connect.Position = [18 118 100 22];
            app.Button_connect.Text = 'Соединить';

            % Create StatusEditFieldLabel
            app.StatusEditFieldLabel = uilabel(app.UIFigure);
            app.StatusEditFieldLabel.HorizontalAlignment = 'right';
            app.StatusEditFieldLabel.Position = [11 463 40 22];
            app.StatusEditFieldLabel.Text = 'Status';

            % Create StatusEditField
            app.StatusEditField = uieditfield(app.UIFigure, 'text');
            app.StatusEditField.HorizontalAlignment = 'center';
            app.StatusEditField.Position = [66 463 239 22];

            % Create Button_disconnect
            app.Button_disconnect = uibutton(app.UIFigure, 'push');
            app.Button_disconnect.ButtonPushedFcn = createCallbackFcn(app, @Button_dis-
connectPushed, true);
            app.Button_disconnect.Position = [134 118 100 22];
            app.Button_disconnect.Text = 'Разъединить';

            % Create DropDown
            app.DropDown = uidropdown(app.UIFigure);
            app.DropDown.Items = {'S11', 'S12', 'S21', 'S22', 'VSWR Port 1', 'VSWR Port 2'};
            app.DropDown.ValueChangedFcn = createCallbackFcn(app, @DropDownValue-
Changed, true);
            app.DropDown.Position = [336 462 100 22];
            app.DropDown.Value = 'S11';

            % Create EditField_StartFreq
            app.EditField_StartFreq = uieditfield(app.UIFigure, 'numeric');
            app.EditField_StartFreq.Limits = [100 6000];
            app.EditField_StartFreq.ValueChangedFcn = createCallbackFcn(app, @Edit-
Field_StartFreqValueChanged, true);
            app.EditField_StartFreq.Position = [134 365 100 22];
            app.EditField_StartFreq.Value = 100;

            % Create EditField_StopFreq
            app.EditField_StopFreq = uieditfield(app.UIFigure, 'numeric');
            app.EditField_StopFreq.Limits = [100 6000];
            app.EditField_StopFreq.ValueChangedFcn = createCallbackFcn(app, @Edit-
Field_StartFreqValueChanged, true);
            app.EditField_StopFreq.Position = [134 322 100 22];
            app.EditField_StopFreq.Value = 1000;

            % Create EditField_PowerLvl
            app.EditField_PowerLvl = uieditfield(app.UIFigure, 'numeric');
```

```
            app.EditField_PowerLvl.Limits = [-26 0];
            app.EditField_PowerLvl.ValueChangedFcn = createCallbackFcn(app, @Edit-
Field_StartFreqValueChanged, true);
            app.EditField_PowerLvl.Position = [134 282 100 22];
            app.EditField_PowerLvl.Value = -10;

            % Create EditField_NumPoint
            app.EditField_NumPoint = uieditfield(app.UIFigure, 'numeric');
            app.EditField_NumPoint.Limits = [2 1001];
            app.EditField_NumPoint.ValueChangedFcn = createCallbackFcn(app, @Edit-
Field_StartFreqValueChanged, true);
            app.EditField_NumPoint.Position = [134 241 100 22];
            app.EditField_NumPoint.Value = 50;

            % Create EditField_RBW
            app.EditField_RBW = uieditfield(app.UIFigure, 'numeric');
            app.EditField_RBW.Limits = [2 140];
            app.EditField_RBW.ValueChangedFcn = createCallbackFcn(app, @But-
ton_measurePushed, true);
            app.EditField_RBW.Position = [134 201 100 22];
            app.EditField_RBW.Value = 2;

            % Create OnlineMeasSwitchLabel
            app.OnlineMeasSwitchLabel = uilabel(app.UIFigure);
            app.OnlineMeasSwitchLabel.HorizontalAlignment = 'center';
            app.OnlineMeasSwitchLabel.Position = [602 431 73 22];
            app.OnlineMeasSwitchLabel.Text = 'Онлайн';

            % Create OnlineMeasSwitch
            app.OnlineMeasSwitch = uiswitch(app.UIFigure, 'slider');
            app.OnlineMeasSwitch.ValueChangedFcn = createCallbackFcn(app, @Online-
MeasSwitchValueChanged, true);
            app.OnlineMeasSwitch.Position = [616 463 45 20];

            % Create Button_Save
            app.Button_Save = uibutton(app.UIFigure, 'push');
            app.Button_Save.ButtonPushedFcn = createCallbackFcn(app, @Button_Save-
Pushed, true);
            app.Button_Save.Position = [18 51 100 22];
            app.Button_Save.Text = 'Сохранить';

            % Create Button_Open
            app.Button_Open = uibutton(app.UIFigure, 'push');
            app.Button_Open.ButtonPushedFcn = createCallbackFcn(app, @Button_Open-
Pushed, true);
            app.Button_Open.Position = [18 30 100 22];
            app.Button_Open.Text = 'Открыть';

            % Create S2PLabel
            app.S2PLabel = uilabel(app.UIFigure);
            app.S2PLabel.Position = [18 72 28 22];
            app.S2PLabel.Text = 'S2P';
```

```
            % Create SmithChartButton_2
            app.SmithChartButton_2 = uibutton(app.UIFigure, 'state');
            app.SmithChartButton_2.ValueChangedFcn = createCallbackFcn(app,
@SmithChartButton_2ValueChanged, true);
            app.SmithChartButton_2.Text = 'Smith Chart';
            app.SmithChartButton_2.Position = [470 463 100 22];

            % Create Button_markers
            app.Button_markers = uibutton(app.UIFigure, 'push');
            app.Button_markers.ButtonPushedFcn = createCallbackFcn(app, @Button_mark-
ersPushed, true);
            app.Button_markers.Position = [703 462 100 22];
            app.Button_markers.Text = 'Маркеры';

            % Create Panel
            app.Panel = uipanel(app.UIFigure);
            app.Panel.Title = 'Маркеры';
            app.Panel.Visible = 'off';
            app.Panel.Position = [569 222 272 165];

            % Create Button_add
            app.Button_add = uibutton(app.Panel, 'push');
            app.Button_add.ButtonPushedFcn = createCallbackFcn(app, @Button_add-
Pushed, true);
            app.Button_add.Position = [20 100 100 22];
            app.Button_add.Text = 'Добавить';

            % Create Button_del
            app.Button_del = uibutton(app.Panel, 'push');
            app.Button_del.ButtonPushedFcn = createCallbackFcn(app, @Button_delPushed,
true);
            app.Button_del.Position = [20 69 100 22];
            app.Button_del.Text = 'Удалить';

            % Create Label
            app.Label = uilabel(app.Panel);
            app.Label.HorizontalAlignment = 'right';
            app.Label.Position = [178 121 80 22];
            app.Label.Text = 'Частота, МГц';

            % Create EditField_freq
            app.EditField_freq = uieditfield(app.Panel, 'numeric');
            app.EditField_freq.Limits = [100 6000];
            app.EditField_freq.Position = [158 100 100 22];
            app.EditField_freq.Value = 100;

            % Create Button_band
            app.Button_band = uibutton(app.Panel, 'state');
            app.Button_band.ValueChangedFcn = createCallbackFcn(app, @But-
ton_bandValueChanged, true);
            app.Button_band.Text = 'Определить';
```

```
app.Button_band.Position = [20 15 100 22];

% Create Label_2
app.Label_2 = uilabel(app.Panel);
app.Label_2.HorizontalAlignment = 'right';
app.Label_2.Position = [206 36 52 22];
app.Label_2.Text = 'Уровень';

% Create EditField_lvl
app.EditField_lvl = uieditfield(app.Panel, 'numeric');
app.EditField_lvl.Position = [158 15 100 22];
app.EditField_lvl.Value = 1;

% Create Label_marker
app.Label_marker = uilabel(app.UIFigure);
app.Label_marker.Visible = 'off';
app.Label_marker.Position = [868 386 56 22];
app.Label_marker.Text = 'Маркеры';

% Create Label_freq
app.Label_freq = uilabel(app.UIFigure);
app.Label_freq.Visible = 'off';
app.Label_freq.Position = [860 355 49 22];
app.Label_freq.Text = 'Частота';

% Create Label_level
app.Label_level = uilabel(app.UIFigure);
app.Label_level.Visible = 'off';
app.Label_level.Position = [941 355 51 22];
app.Label_level.Text = 'Уровень';

% Create Label_marker_1
app.Label_marker_1 = uilabel(app.UIFigure);
app.Label_marker_1.Position = [860 323 248 23];
app.Label_marker_1.Text = '';

% Create Label_marker_2
app.Label_marker_2 = uilabel(app.UIFigure);
app.Label_marker_2.Position = [860 298 248 22];
app.Label_marker_2.Text = '';

% Create Label_marker_3
app.Label_marker_3 = uilabel(app.UIFigure);
app.Label_marker_3.Position = [860 273 248 22];
app.Label_marker_3.Text = '';

% Create Label_marker_4
app.Label_marker_4 = uilabel(app.UIFigure);
app.Label_marker_4.Position = [860 248 248 22];
app.Label_marker_4.Text = '';

% Create Label_marker_5
```

```
            app.Label_marker_5 = uilabel(app.UIFigure);
            app.Label_marker_5.Position = [860 223 248 22];
            app.Label_marker_5.Text = '';

            % Create Label_marker_6
            app.Label_marker_6 = uilabel(app.UIFigure);
            app.Label_marker_6.Position = [860 199 248 22];
            app.Label_marker_6.Text = '';

            % Create Switch
            app.Switch = uiswitch(app.UIFigure, 'slider');
            app.Switch.Items = {'Mag', 'Phase'};
            app.Switch.ValueChangedFcn = createCallbackFcn(app, @SwitchValueChanged,
true);
            app.Switch.Position = [363 432 45 20];
            app.Switch.Value = 'Mag';

            % Create UIAxes
            app.UIAxes = uiaxes(app.UIFigure);
            xlabel(app.UIAxes, 'freq, Hz')
            ylabel(app.UIAxes, 'dB')
            app.UIAxes.PlotBoxAspectRatio = [1.47567567567568 1 1];
            app.UIAxes.XGrid = 'on';
            app.UIAxes.YGrid = 'on';
            app.UIAxes.Position = [268 9 593 424];

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = app1

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
```

```
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

ПРИЛОЖЕНИЕ Г

Доработанные методические указания для выполнения лабораторных работ

Данные методические указания являются доработкой методических указаний «Устройства СВЧ и антенны» [5].

Лабораторная работа №3

Описание стенда с микрополосковой линией

Используемый лабораторный стенд имеет габариты 460×180×60 мм (рисунок Г.1). В его состав входят:

– подложка с двумя слоями диэлектрика (слой стеклотекстолита у токопроводящей полоски толщиной 1 мм и воздушная прослойка между ним и экраном толщиной 2 мм);

– металлическое шасси, выполняющее роль экрана микрополосковой линии (МПЛ);

– металлизированная полоска шириной 11 мм и толщиной 35 мкм, выполняющая роль токопроводящей полоски МПЛ;

– два высокочастотных разъема типа *N*‑*female*;

– линейка для отсчета расстояний от нагрузки.

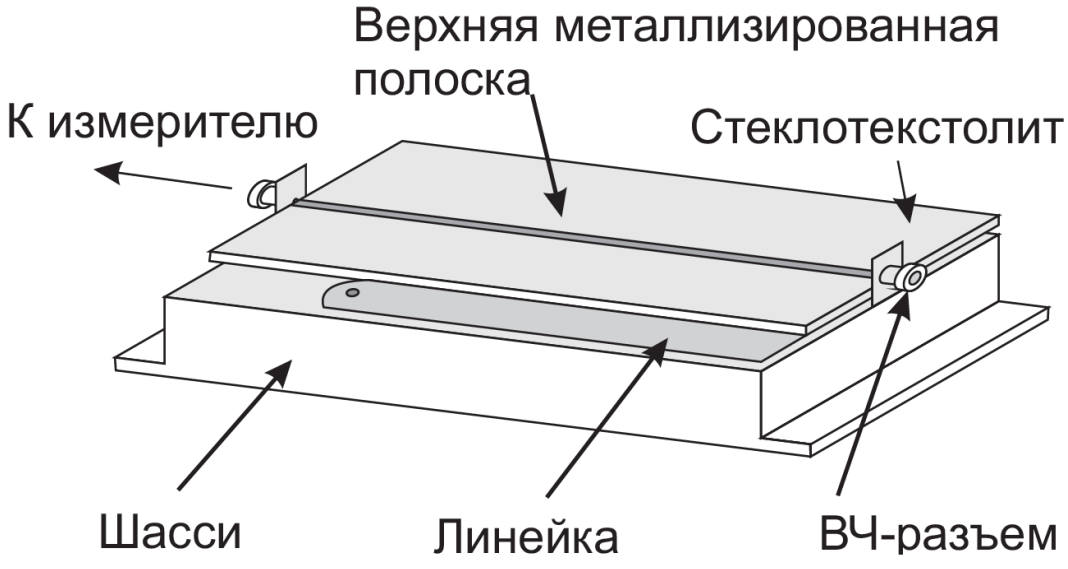


Рисунок Г.1 – Стенд с полосковой линией передачи

Ширина верхнего проводника, толщина диэлектрических слоев и их диэлектрическая проницаемость подобраны таким образом, чтобы волновое сопротивление МПЛ было равно ρ = 50 Ом. Стенд при помощи переходника *N-Male* на *SMA-Female* через один ВЧ-разъем подключается к измерителю, на другой выход стенда подключается нагрузка.

Для измерения импедансных характеристик нагрузки в работе используется модульный комплект векторного анализатора цепей СВЧ фирмы *Mini – Circuits UVNA*-63. Подготовка к работе прибора производится преподавателем.

Для изготовления элементов согласования в учебной лаборатории используется алюминиевая фольга с клеевым слоем (металлизированная клейкая лента). По расчетным размерам вырезаются полоски (прямоугольники) и наклеиваются на основную линию для получения согласующих элементов.

Экспериментальная часть

Предлагается выполнять эксперимент в соответствии с вариантами задания, указанными в таблице 1, в следующем порядке.

1. К рабочему стенду с полосковой линией передачи подключить исследуемую нагрузку в соответствии с вариантом задания. Определить комплексное нормированное сопротивление нагрузки по номограмме на указанной частоте. Определить КСВ.

2. Определить двумя методами – графическим и аналитическим – относительную длину $l_{хх}/\lambda_{л}$ согласующего параллельного шлейфа ХХ и расстояние до его места установки $\Delta l/\lambda_{л}$. Рассчитать длину $l_{хх}$ и расстояние Δl, используя график на рисунке Г.3.

3. Вырезать из фольги шлейф, наклеить его на рабочем поле стенда на расчётном расстоянии $l_{хх}$ от нагрузки. Расстояние до точки включения шлейфа ХХ откладывается по прикрепленной к стенду линейке, от точки «0» на линейке до центра шлейфа.

4. Измерить нормированное комплексное сопротивление нагрузки с согласующим элементом. Определить КСВ. В случае если значение КСВ >1,2,

оптимизировать размеры шлейфа, подрезав или доклеив проводящие полоски. Записать получившиеся размеры и положение.

5. Определить двумя методами – графическим и аналитическим – параметры согласующего четвертьволнового трансформатора ($\Delta l/\lambda_л$, $\rho_{тр}$). Используя графики на рисунке Г.2 и рисунке Г.3, рассчитать длину трансформатора $\lambda_л/4$, расстояние до точки включения Δl, ширину полоски трансформатора $w_{тр}$.

6. Вырезать из фольги трансформатор и наклеить на рабочем поле стенда на расчетном расстоянии Δl от нагрузки. Расстояние до точки включения трансформатора откладывается по прикрепленной к стенду линейке, от точки «0» на линейке до начала (ближней к нагрузке точки) трансформатора.

7. Измерить нормированное комплексное сопротивление нагрузки с четвертьволновым трансформатором. Определить КСВ. В случае если значение КСВ >1,2, оптимизировать размеры шлейфа, записать получившиеся размеры и расстояние от нагрузки.

Таблица 1 – Варианты заданий

Номер задания	1	2	3	4	5	6
Номер нагрузки	1	2	3	4	1	2
Частота, МГц	900	900	900	900	900	900

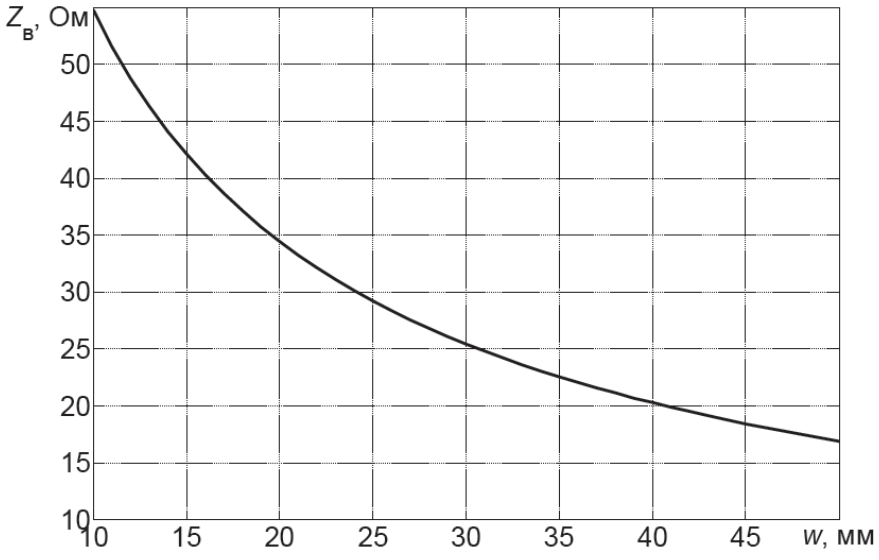


Рисунок Г.2 – Зависимость волнового сопротивления ρ (или $Z_в$) микрополосковой ЛП от ширины полоски *w*

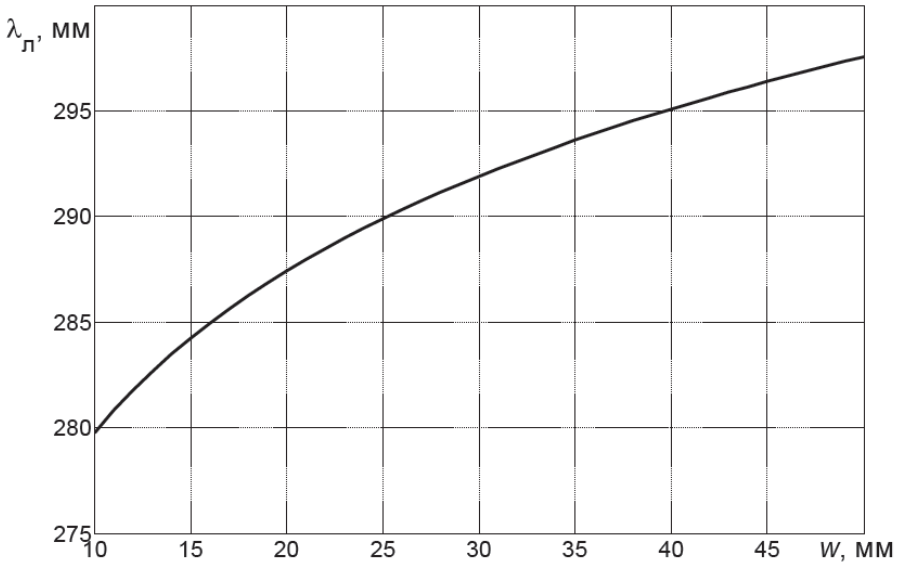


Рисунок Г.3 – Зависимость длины волны $\lambda_л$ в ЛП на частоте f = 900 МГц от ширины полоски w

Лабораторная работа №4

Подготовка к эксперименту

В эксперименте исследуются АЧХ ППФ, ПЗФ, ФНЧ, реализованных на микрополосковых линиях. Для снятия частотных характеристик используется измеритель комплексных коэффициентов передачи и отражения.

Лабораторный стенд состоит из двухслойной диэлектрической подложки с ВЧ-разъемами типа N по бокам для подключения измерителя комплексных коэффициентов передачи и отражения «*Mini – Circuits UVNA*-63». В качестве измеряемого параметра выступает коэффициент передачи по напряжению S_{21}. На рисунке Г.4 изображена схема подключения измерителя к ПК. Подключенное измерительное устройство выводит данные через *USB*-кабель на персональный компьютер.

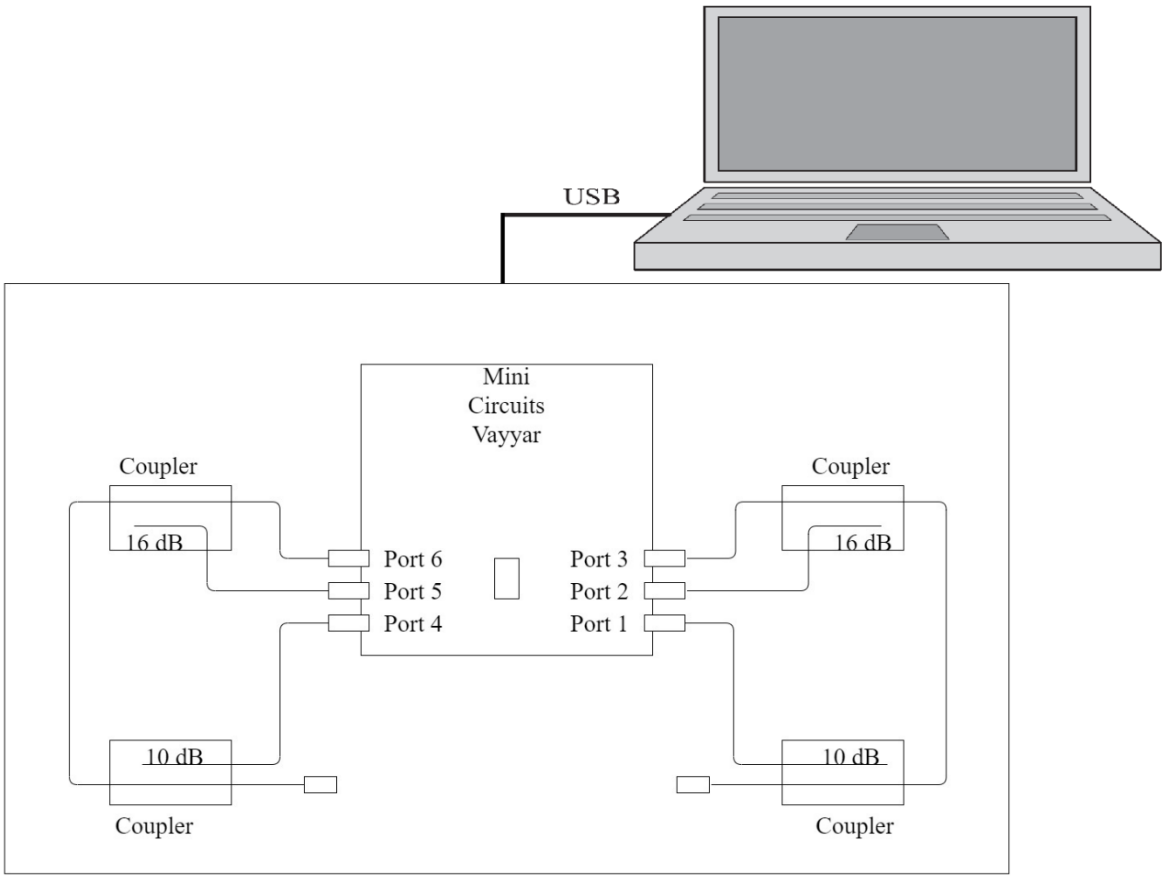


Рисунок Г.4 – Состав и схема лабораторной установки

Для подготовки установки к работе произведите следующие действия.

1 Соедините прибор «*Mini – Circuits UVNA*-63» с персональным компьютером кабелем *USB*. Соберите схему измерения, изображенную на рисунке Г.5, подключив к измерительным портам коаксиальные кабели, соединенные с исследуемым устройством.

2 Запустите программу «*UVNA*-63». Установите диапазон частот от 0,3 ГГц до 1,3 ГГц. В графе «*Start Freq*» установите частоту 300 МГц, а в графе «*Stop Freq*» 1300 МГц. Завершите ввод нажатием на клавиши «*Enter*».

3 Произведите калибровку измерителя. Калибровка необходима для выставления корректного модуля и фазы коэффициента передачи с учетом соответственно затухания и набега фазы в линии стенда и соединительных кабелях. Калибровка позволяет правильно установить плоскости отсчета параметров исследуемого устройства. Она требует, с помощью калибровочного набора, подключения к порту 1 и 2 нагрузки холостого хода, короткозамкнутой нагрузки и согласованной нагрузки после этого портов 1 и 2 друг к друг. Вызовите команду «Калибровка», расположенного в нижней части рабочего окна. При этом рассчитается таблица калибровочных коэффициентов и они будут автоматически применены к результатам измерений.

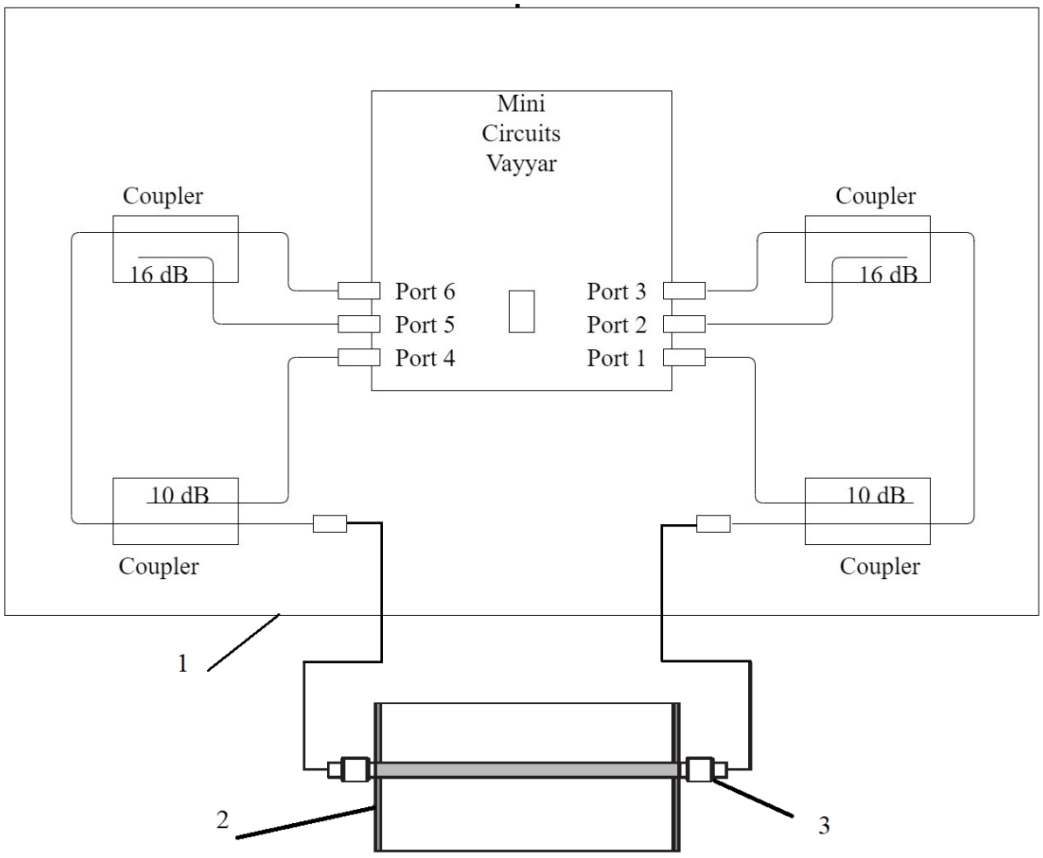
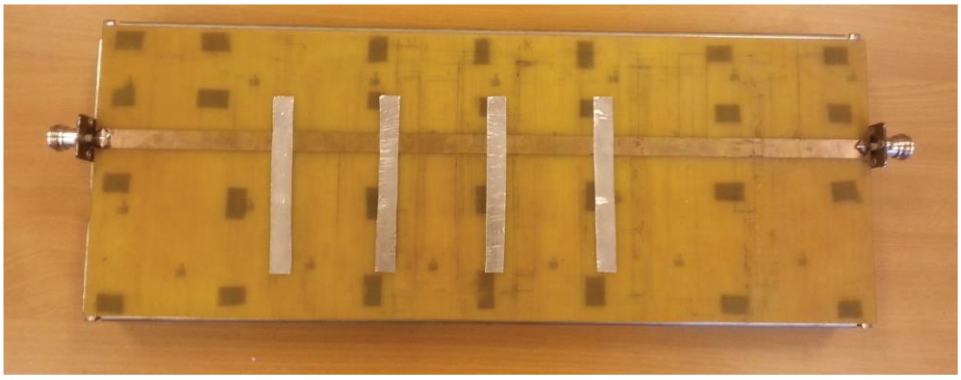


Рисунок Г.5 – Схема соединения стенда с измерителем при калибровке: 1 — прибор «*UVNA*-63»; 2 — лабораторный стенд; 3 — измерительные кабели

4. Прибор готов к измерению. Для измерения характеристик фильтров необходимо элементы фильтра вырезать из фольги и наклеить на рабочий стенд с микрополосковой линией передачи. При наклеивании отрезка на центральную линию необходимо перекрыть ее ширину. Нанесите по эскизам домашнего задания топологию фильтров на диэлектрическую подложку. Пример наклеенных отрезков фильтра представлен на рисунке Г.6.



Рисунке Г.6 – Пример нанесения полосок на диэлектрическую подложку

Экспериментальная часть

При помощи маркеров определить полосу частот пропускания/заграждения по уровням –3 дБ, –15 дБ, при необходимости добавить дополнительные маркеры в других характерных точках АЧХ для большей информативности. Маркеры – это инструмент для считывания числовых значений стимула и измеряемой величины на выбранных точках графика. Измеритель позволяет включать до 5 маркеров на график. Вид графика с двумя маркерами показан на рисунке Г.7. Для установки нового маркера в окне «Маркеры > Частота» установите нужную частоту и нажмите программные кнопки «Маркеры > Добавить». Новый маркер устанавливается на нужной частоте.

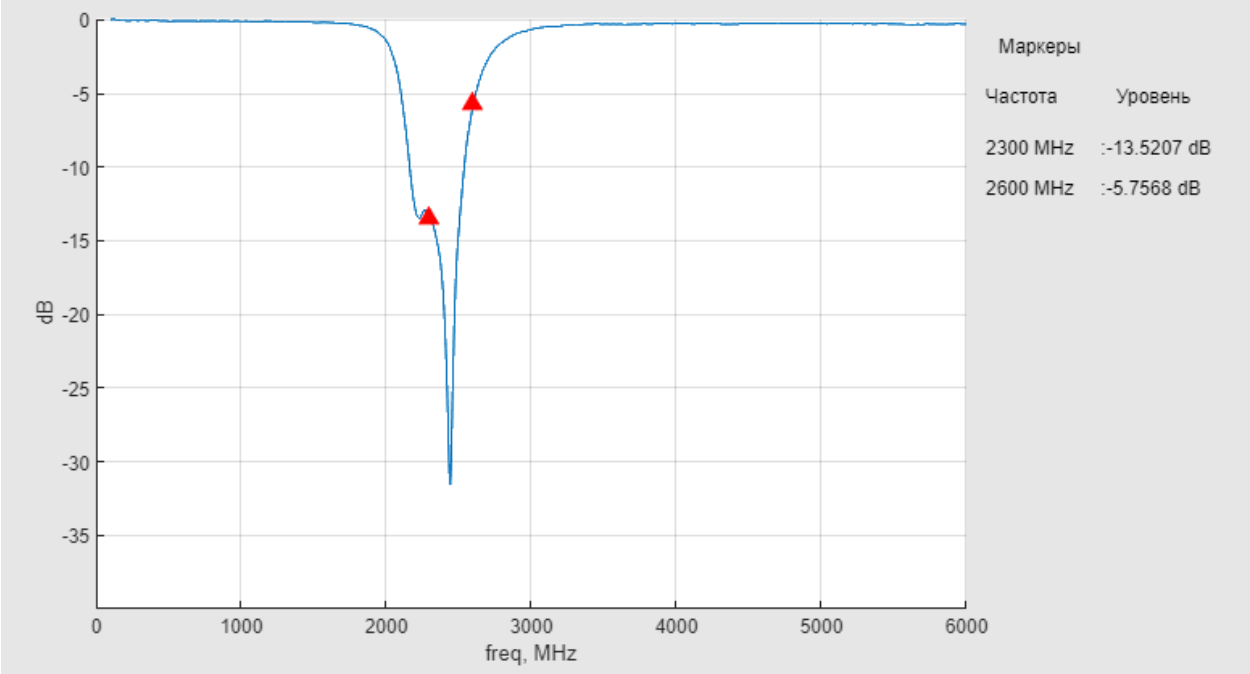


Рисунок Г.7 – Маркеры на измеренных зависимостях

Для удаления маркера нажмите программные кнопки «Маркеры > Удалить».

Используя функцию определения полосы, можно находить положения маркеров, соответствующих началу полосы и концу полосы. Для определения полосы нужно в окне «Маркеры > Уровень» установить нужный уровень и нажать кнопку «Маркеры > Определить». Вид графика с маркерами в начале и конце полосы изображен на рисунке Г.9.

Графики могут быть записаны в файлы в виде таблиц либо растровых изображений. Для сохранения графика в виде изображения необходимо сверху графика в выпадающем списке нажать кнопку «*Save as*» и в открывшемся диалоговом окне указываются имя сохраняемого файла и путь. Далее нужно нажать «Сохранить»

Для сохранения графика нужно вверху окна перейти в меню «Файл > Сохранить» и выбрать нужный из форматов для сохранения измеренных данных.

После выбора формата файл с данными сохраниться в корневом каталоге в папке «*measure*», с именем «*dd mmm yyyy hh-mm-ss -measure*» с расширением «*.S2P*» или «*.csv*». Пример сохранения данных изображен на рисунке Г.8.

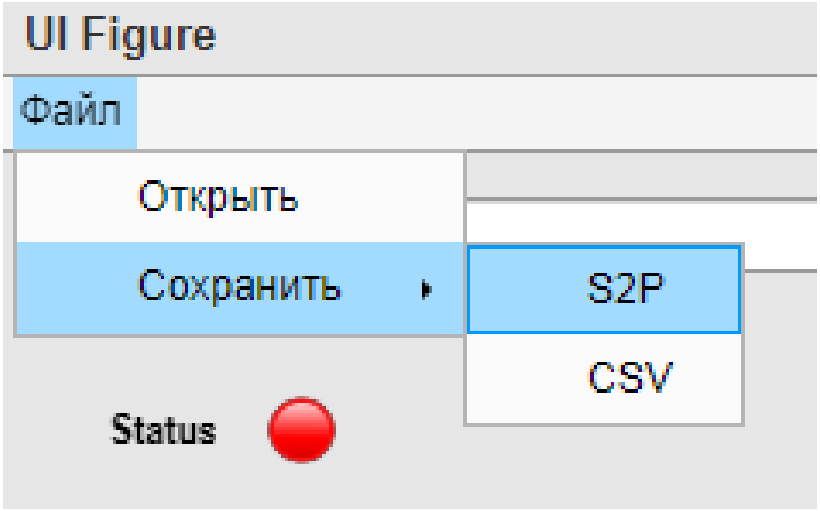


Рисунок Г.8 – Пример сохранения графика

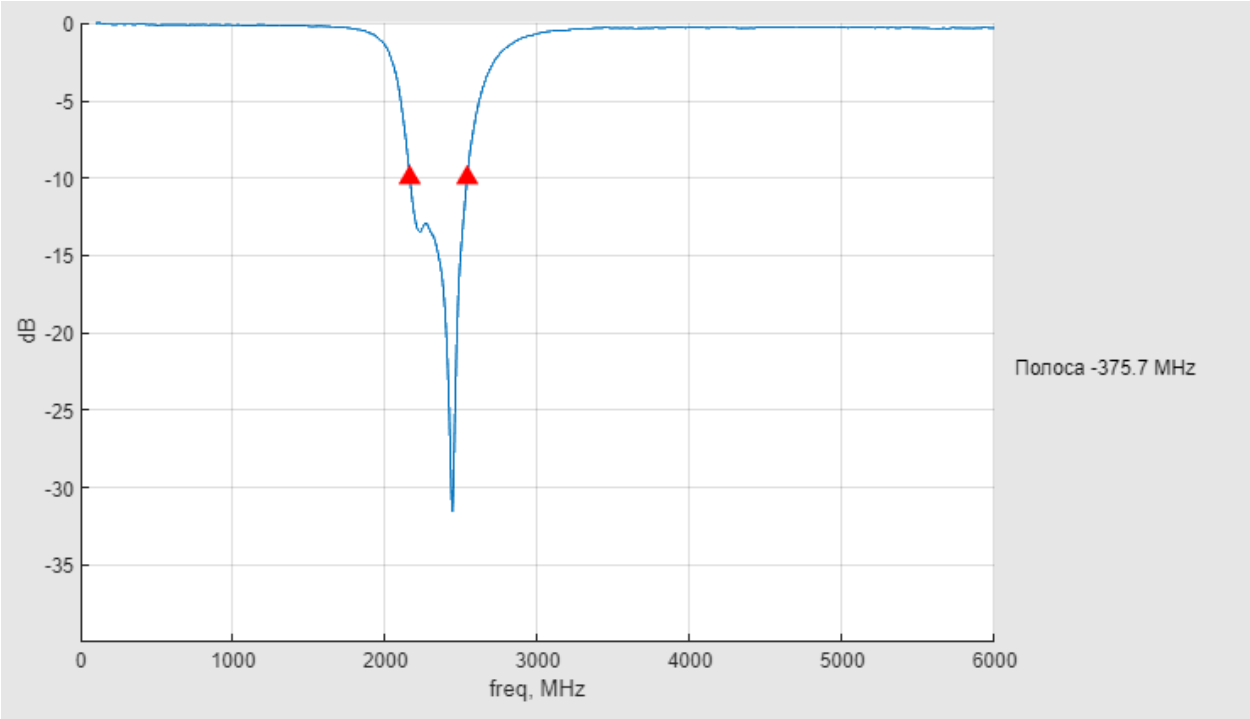


Рисунок Г.9 – Установка маркеров значений полосы на графике

Снимите АЧХ фильтра. По данным измерений определите основные параметры фильтров $f_п$, $f_з$ (по уровню $L_з$ = –15 дБ), $L_п$.

Лабораторная работа №8

Исследуются импедансные характеристики одиночных полосковых излучателей и ДН полосковых АР. Схема измерений приведена на рисунке Г.10. Перед началом работы измерительное устройство «*Mini – Circuits UVNA*-63» подключается к *USB*-порту компьютера. Далее необходимо запустить приложение «*UVNA*-63».

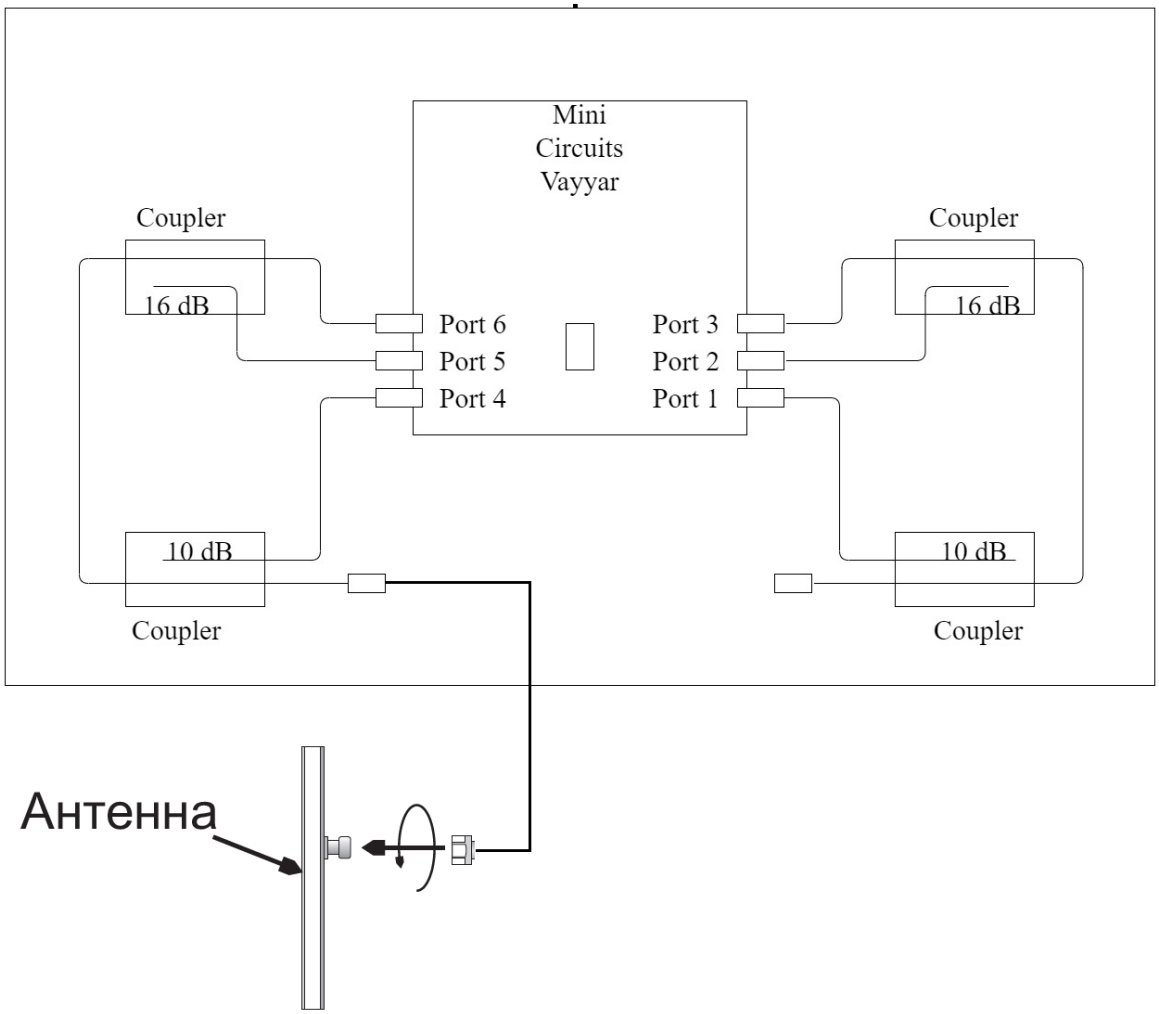


Рисунок Г.10 – Схема лабораторной установки

Подготовка к измерениям. Перед проведением сеанса измерений рекомендуется откалибровать аппарат. Для калибровки нажмите кнопку «Калибровка», расположенную в нижней части рабочего окна. . После нажатия на кнопку программа перейдет в консольное окно программы *Matlab* и начнется калибровка. Пользователю предстоит провести 7 измерений. Последовательно

подключать калибровочные меры сначала к одному порту потом к другому. Пример калибровки изображен на рисунке Г.11.

```
>> Measure OPEN @ Port-1:
Recording...Done.

>> Measure SHORT @ Port-1:
Recording...Done.

>> Measure LOAD @ Port-1:
Recording...Done.

-- Switch to Port-2 --

>> Measure OPEN @ Port-2:
Recording...Done.

>> Measure SHORT @ Port-2:
Recording...Done.

>> Measure LOAD @ Port-2:
Recording...Done.

>> Measure THRU:
Recording...Done.

Constructing 12-Term Error Model...
```

Рисунок Г.11 – Процедура двухпортовой калибровки *SOLT*

При этом рассчитается таблица калибровочных коэффициентов и они будут автоматически применены к результатам измерений.

Установите диапазон частот от 0,3 ГГц до 1,3 ГГц. В графе «Start Freq» установите частоту 300 МГц, а в графе «Stop Freq» 1300 МГц. Завершите ввод нажатием на клавиши «Enter». Пример показан на рисунке Г.12.

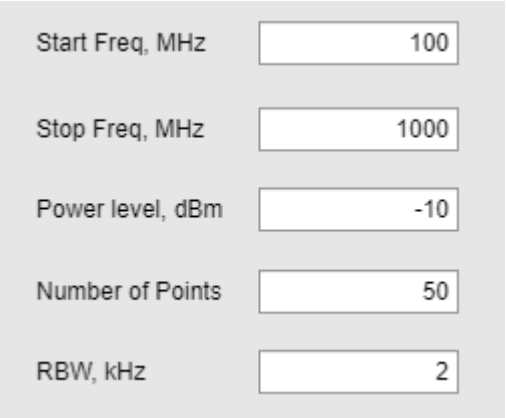


Рисунок Г.12 – Поля параметров измерения

Изначально параметры стимулирующего сигнала, следующие: диапазон частот от 85 МГц до 5,4 ГГц, закон сканирования по частоте — линейный, число точек — 201, уровень выходной мощности — -10 *dBm*, полоса *RBW*— 10 кГц. Для установки параметров стимулирующего сигнала введите их в соответствующие окна на экране.

Установите диапазон измерения 750–1100 МГц. Для этого введите на цифровой клавиатуре нижнюю частоту для вывода графиков. Завершите ввод нажатием на кнопку «*Enter*». Аналогично введите верхнюю частоту в поле над параметром.

Для выбора другого графика или изменения параметров существующего графика, вверху главного окна из выпадающего списка нужно выбрать «*VSWR Port* 1». Выбор типа графика изображен на рисунке Г.13.

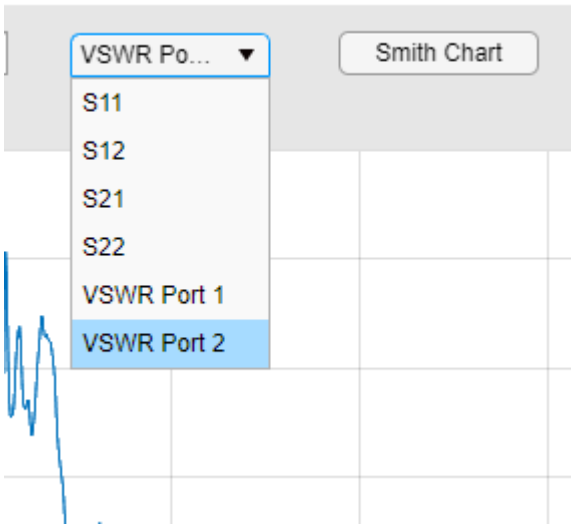


Рисунок Г.13 – Выбор типа графиков

Для удобства работы масштаб графиков изменяется с помощью функции автомасштабирования.

Проведение измерений. Подключите антенну к прибору. Для определения центральной частоты полосы согласования антенны необходимо установить маркер в точке минимума графика. Для установки нового маркера в окне «Маркеры > Частота» установите нужную частоту и нажмите программные кнопки «Маркеры > Добавить». Новый маркер устанавливается на нужной частоте. Определение 2 маркеров изображено на рисунке Г.14.

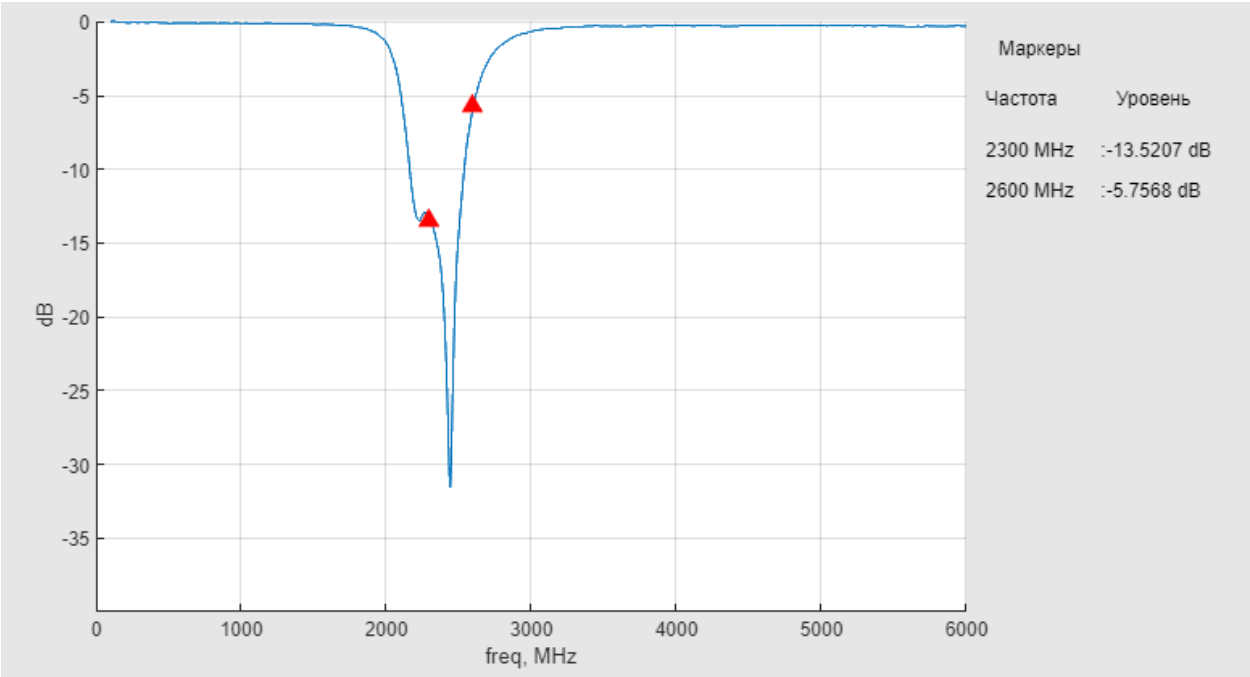


Рисунок Г.14 – Маркеры на измеренных зависимостях

Для удаления маркера нажмите программные кнопки «Маркеры > Удалить».

Используя функцию определения полосы, можно находить положения маркеров, соответствующих началу полосы и концу полосы. Для определения полосы нужно в окне «Маркеры > Уровень» установить нужный уровень и нажать кнопку «Маркеры > Определить». Вид графика с маркерами в начале и конце полосы изображен на рисунке Г.15.

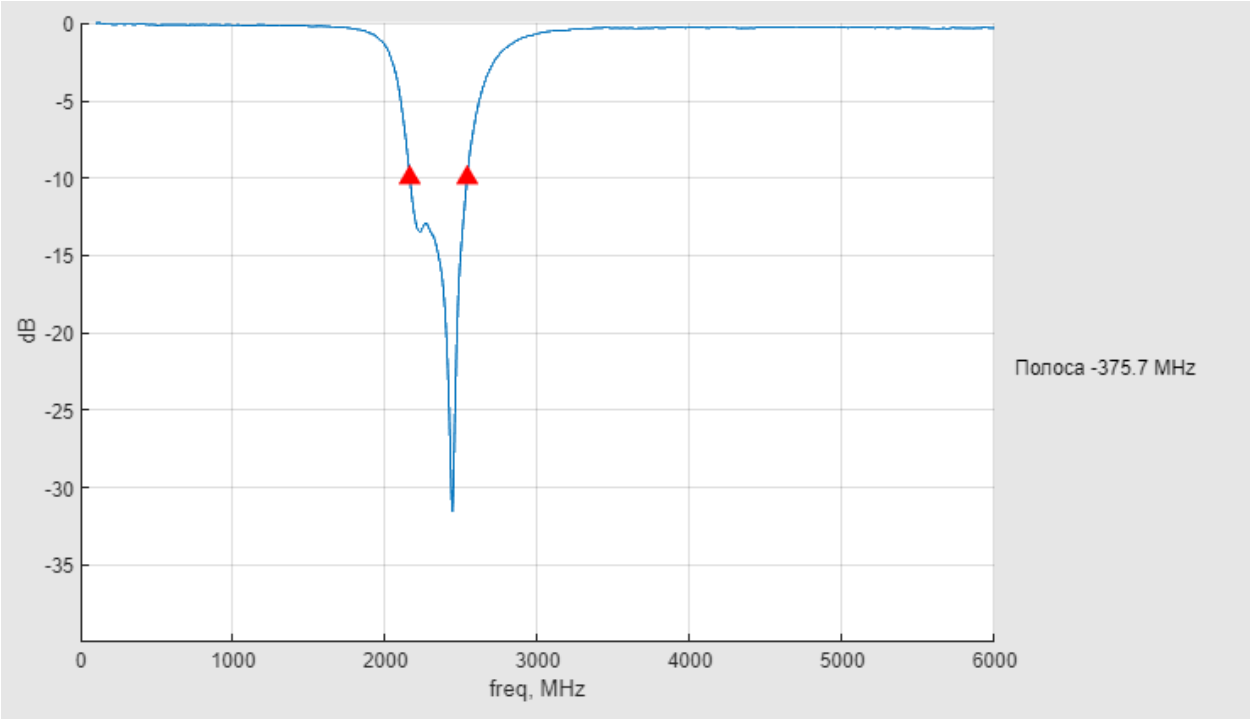


Рисунок Г.15 – Установка маркеров значений полосы на графике

Графики могут быть записаны в файлы в виде таблиц либо растровых изображений. Для сохранения графика в виде изображения необходимо сверху графика в выпадающем списке нажать кнопку «*Save as*» и в открывшемся диалоговом окне указываются имя сохраняемого файла и путь. Далее нужно нажать «Сохранить»

Для сохранения графика нужно вверху окна перейти в меню «Файл > Сохранить» и выбрать нужный из форматов для сохранения измеренных данных.

После выбора формата файл с данными сохраниться в корневом каталоге в папке «*measure*», с именем «*dd mmm yyyy hh-mm-ss -measure*» с расширением «*.S2P*» или «*.csv*». Пример сохранения данных изображен на рисунке Г.16.

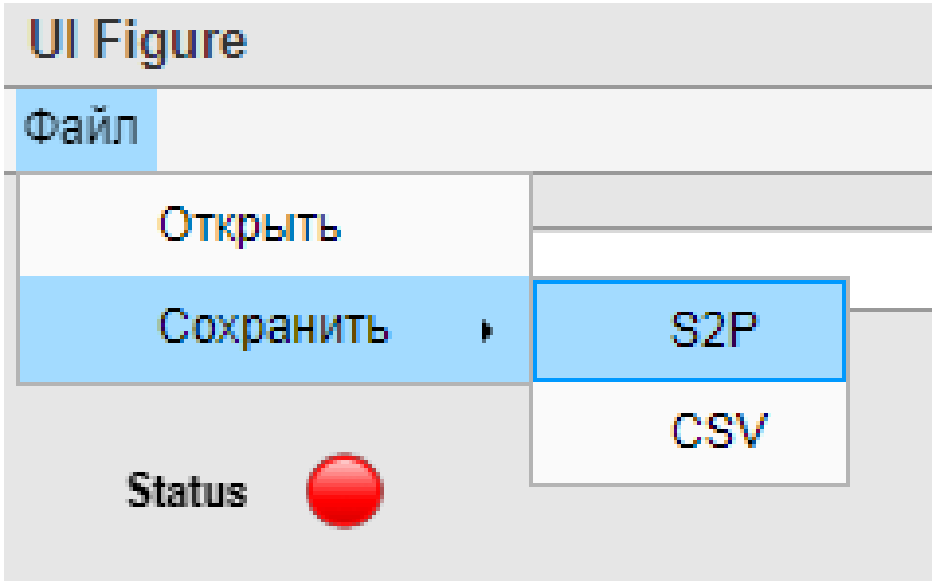


Рисунок Г.16 – Пример сохранения графика

Измерение диаграммы направленности проводится по схеме, представленной на рисунке Г.17. В качестве передающей антенны используется измерительная рупорная антенна П6–23 А. Рупор вращается вокруг продольной оси, что используется для смены поляризации или снятия характеристик передающей антенны.

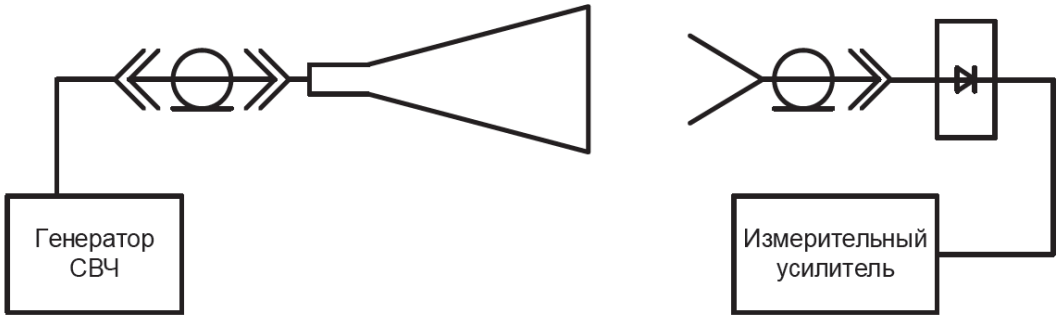


Рисунок Г.17 – Схема измерения ДН

Экспериментальная часть

1. Используя схему измерений, показанную на рисунке Г.17, для излучателей № 1, 2, 5 подобрать там, где это возможно, оптимальные по согласованию положения точек питания. Определить их резонансные частоты, полосы согласования по уровню КСВ ≤ 2 и уровень КСВ на центральной частоте полосы согласования.

2. Для излучателей № 3, 8, 9, 10, 11 измерить ширину полосы согласования по уровню КСВ ≤ 2. Сравнить измеренные полосы согласования.

3. Для излучателя № 4 провести измерения резонансных частот и уровня КСВ на центральной частоте полосы согласования при углах α = 0°, 45°, 90°, в каждом случае определить поляризацию излучаемого поля. Для этого необходимо установить антенну в качестве приемной на лабораторном стенде. Далее, вращая передающую рупорную антенну вокруг ее оси, необходимо зафиксировать положение, при котором наблюдается максимальный сигнал на приемнике (максимум коэффициента передачи между антеннами). Записать угол поворота по шкале на рупорной антенне.

4. Для излучателя № 6 определить резонансную частоту, уровень КСВ на центральной частоте полосы согласования, поляризацию излучаемого поля по методике п. 3.

5. Снять ДН антенных решеток (схема измерений на рисунке Г.17) на частоте f_0. Значения f_0 указаны на антенных решетках. Учесть квадратичность характеристики детектора, используя формулу:

$$\mathrm{F}(\theta) = \sqrt{\mathrm{U}(\theta) \,/\, \mathrm{U}_{max}},$$

где $\mathrm{U}(\theta)$ – показания индикатора усилителя,

U_{max} – максимальный уровень сигнала.