1603St-Regis
D.D.O., Quebec H9B 3H7, Canada
Tel 514-684-4554 Fax 514-684-8581
E-mail: info@ focus-microwaves.com
Website: http://www.focus-microwaves.com

FOCUS GPTC Library
Software Description and Programming Instructions

Introduction

The FOCUS GPTC (General Purpose Tuner Controller) library is made for users who want to create their own automated load pull test application using either high level instrumentation languages like LabView™ from National Instruments or HP-VEE™ from Agilent, or scientific software like MATLAB™ or IGOR™ from Matrix Waves. For this purpose different interfaces are available form FOCUS Microwaves: different libraries are available for Microsoft Visual C++ 6.0 (GPTCa.LIB), Borland Builder (GPTCb.LIB) and WATCOM C++ (GPTCc.LIB).
Recently, FOCUS Microwaves added support to ActiveX with a COM interface (xGPTC.c).

Library demo distribution

The FOCUS demo library distribution (FOCUS_DEMO.ZIP) includes the following:
- 3 library files: FOCUSPCI_LIB.LIB, GPIBNAT.LIB and GPTCa.LIB
- xGenericTuner.h, which includes all the functions that are available in IGOR

2 tuner calibration files are included T1201620.CAL (for the load tuner) and T2201620.CAL (for the source tuner). In FOCUS nomenclature Tuner 1 (T1) is the load tuner and Tuner 2 (T2) is the source tuner.
Note that these calibration files are demonstration files not specifically made for your tuners, which shall be calibrated independently for actual measurements.

Also two S2P files describing the test fixture (FOCUS' coaxial power transistor Minimum Loss Test Fixture, MLTF) between the tuners and DUT are included.
The names of these two files are: CONA.S2P and CONB.S2P (for Connector A and Connector B).
You can add as many S2P files as you want to describe your networks.
The files are internally cascaded by the software when you add them so that the actual Smith Chart displays always impedances and losses at DUT reference plane.

A tuner control and test example, FOCUS.EXE, compiled using tuner control and display functions of the GPTC library using Microsoft Visual C++ 6.0 is also included, in order to help Users understand how the library functions work.
The associated project files are: focus.dsw, focus.dsp, focus.cpp, focus.h, focus.rc, ressource.h, StdAfx.h and StdAfx.cpp.

 June 2002

Tuner description

In your program before you are able to use a tuner you have to create it as the load, source or inter-stage tuner by calling the appropriate method:

```
tunerHandle create_source_tuner(void);
tunerHandle create_load_tuner(void);
```

This creates a handle that will support your source, load or inter-stage network.

Source tuner (*includes all components from the source to the DUT*):
The setup description should go from the signal source (which should be at 50Ω) to the DUT. The description includes your input coupler and circulator and (optional) bias-tee* modeled by S2P files, followed by your tuner configuration, typically a fundamental tuner (CCMT, Computer Controlled Microwave Tuner or PMT, Prematch Tuner), and a harmonic tuner (PHT). Then you have to finish the description of your input network with the S2P file describing the input half of the test fixture (in our case the MLTF, but any fixture or wafer probe configuration can be used). Since this is a cascaded network, it is important to go from the source to the DUT when entering the configuration. To complete the description of the different networks, you have to call:

```
BOOL add_s2p_file(tunerHandle hTuner, char* fileName);
BOOL add_ccmt(tunerHandle hTuner, long card, long portIndex, long
xMax, long yMax, double stepSize);
```

And similarly: `add_pmt(), add_pht(), add_combo()`**.

Load tuner *(includes all components from the DUT to the load*)**:**
This is almost the same as for the source side of the setup, from the DUT, the output half of the test fixture, the output tuners, bias-tee and attenuators to the load, using the same functions as for the source side. Again the description is supposed to end in a 50Ω load.

Tuner initialization

The first thing to be done, once created, is to initialize your tuner

```
BOOL initialize(tunerHandle hTuner);
```

This will move the tuner carriage and probe to the zero position, whih is up and left for the output tuners and up and right for the input tuners.

** Bias networks may be included on the test fixture*
*** Combo tuners are units allowing manipulation of fundamental and harmonic frequencies in a single unit.*

Moving X, Y

After your setup is created, you can use different methods to move the slugs or harmonic resonators.

```
BOOL set_tuner_position(tunerHandle hTuner, long x, long y);
```

This will move the fundamental tuner probe (slug) to an x, y position of CCMT, PMT or COMBO tuner.
Similarly `Set_prematch_position()` will move the prematch section of a PMT to an x, y position and
`Set_resonator_1_position()` and `Set_resonator_2_position()` will move the different resonators of harmonic (PHT) or COMBO tuners.

Loading and using Tuner Calibration files

Once a tuner is characterized (calibrated) on a VNA using FOCUS WinCCMT (TWIN) software you can specify and use the calibration file of your tuner to your setup in order to display and tune accurately to impedances over the entire Smith Chart at the corresponding frequency.
The corresponding command is:

```
BOOL set_calibration_file(tunerHandle hTuner, char* fileName);
```

Where fileName is the name of the tuner calibration file, like c:\focus\cal\example\T1402000.CAL

S-parameters of your setup

The 4 S-parameters of your setup between signal source and DUT are included in the source tuner and between DUT and load in the load tuner.
They can be retrieved at all positions of the tuners using:

```
BOOL get_sij_parameter(tunerHandle hTuner, long freq_mhz, long
sij, double *sij_mag, double *sij_phase);
```

Tuning

This set of functions represents the very basic functionality of the different tuners. If you have a calibration file for the tuner (.CAL file, loaded with the `set_calibration_file()` function), you then have access to the tuning functions. In the DUT reference plane, you can tune the source and load impedances or gamma. The phase of the harmonic tuners can also be adjusted.
The functions are:

```
BOOL set_impedance(tunerHandle hTuner, long freq_mhz, double
z_real, double z_imag);
BOOL set_gamma(tunerHandle hTuner, long freq_mhz, double
gamma_magnitude, double gamma_phase);
```

Note: in demo mode, the tuning is limited to calibration points from the cal file. So, when you request any other impedance it will jump to the closest calibration point.

Pattern

"Square", "Circular" and "Pie (wedge)" patterns on the Smith chart can be defined using:

```
BOOL define_square_pattern(tunerHandle hTuner, double
lt_gamma_real, double lt_gamma_imag, double rb_gamma_real, double
rb_gamma_imag, long nbMinPoints, BOOL is_concentric);
// lt = Left Top, rb = Right Bottom

BOOL define_circle_pattern(tunerHandle hTuner, double
gamma_magnitude, long nbMinPoints, BOOL is_concentric);

BOOL define_wedge_pattern(tunerHandle hTuner, double
gamma_min_magnitude, double gamma_max_magnitude, double alpha_min,
double alpha_max, long nbMinPoints, BOOL is_concentric);
```

Different distributions of points are available. The different patterns can be displayed on the Smith Chart available using the display function.

Two functions allow you to iterate (=access one by one) through these sets of points:

```
BOOL tune_to_first_impedance(tunerHandle hTuner);
BOOL tune_to_next_impedance(tunerHandle hTuner);
```

When reaching the end of the set, `goto_next_impedance()` returns FALSE.
This makes it easy to everyone to quickly test their DUT at defined impedances. The value of the impedance at each step can be retrieved with `get_impedance()`.

Note: in demo mode, the tuning is limited to calibration points from the cal file.

Display

In order to allow RF test engineers to access and visualize fast their work, FOCUS GPTC library provides a Smith Chart of the tuner impedances. The command is:

```
void display(tunerHandle hTuner, BOOL on, HWND hWndParent);
```

The display menu can be accessed by right clicking the mouse inside the corresponding tuner.
- display the calibration points
- display the pattern you want to measure
- zoom in / zoom out into the Smith window
- sliding bars: you can move the 2 motors with the sliding bars. Increment with step 10 when

pressing the arrow key and increment of 1 motor step while holding the CTRL key in combination with the arrow key.

Saving Load Pull data in .LPD (Load Pull Data) files

You can easily save your measurement data into .LPD files using:

```
BOOL create_lpd_file(tunerHandle hTuner, char* fileName, char*
header, char* column_titles);
BOOL add_data_to_lpd_file(tunerHandle hTuner,int size,double* d);
```

and visualize them using the graphical contouring software package “Surfer” version 8 provided with the Focus Software.