Application Note

AEROFLEX
A passion for performance.

Simultaneous Transmitter and Receiver Calibration Using PXI 3000 Fast Sequence Tuning Software

This application note describes how to use Aeroflex PXI 3000 Series RF modular instruments and Fast Sequence Tuning (FST) software to perform high speed simultaneous alignment of 2G and 3G mobile transmitters and receivers.

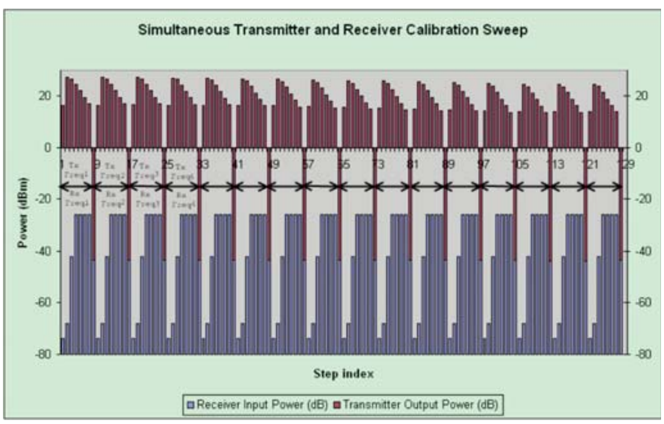
Simultaneous transmitter and receiver calibration

Mobile phones require alignment during manufacturing to compensate for level and frequency response. Alignment requires multiple measurements to be performed under various test conditions.

To speed up alignment, many modern mobile devices can transmit a predefined sequence of power levels on different frequency channels in a single sweep. Some mobile devices can also simultaneously receive corresponding downlink signals from which Rx alignment can be performed at the same time.

Using the Aeroflex Fast Sequence Tuning measurement suite, it is possible to easily control PXI 3000 Series Modular RF Instruments to supply the required downlink signal and measure the device's key performance indicators, such as output power, for each step.

Below is an illustration of a typical simultaneous transmitter and receiver calibration sweep. The red trace (top) shows the device's transmitter output powers; the blue trace (bottom) displays the receiver's input power levels.



In this example, the device transmits seven power levels (16, 27, 26, 24, 22, 19, 16 dBm) and repeats these power levels on a total of 16 frequency channels. You may notice that in every eighth step, the transmit power is very low. This is because the device uses this period to re-tune its frequency to the next channel.

While the instrument captures and measures the power of each step, the signal generator also provides a downlink signal at a series of different power levels (-74, -68, -26, -26, -26, -26, -26, -26 dBm) for the device to calibrate its receiver. These power levels are also repeated on a total of 16 frequency channels. Because PXI 3000 RF modules can switch frequency much faster than the mobile phone it is possible to be settled on the new frequency in advance of the mobile phone making each measurement.

Key benefits of the FST measurement suite are:

- Supports up to 128 different frequency channels across any band in a single sweep
- Flexible number of power level steps per frequency channel (subject to PC and digitizer memory size and downlink signal levels)
- Fully configurable step size, from GSM time slot width (577 μs) to WCDMA 20 ms or even longer (subject to PC and digitizer sample memory size)
- Analyze power, phase and frequency error
- Provides IQ data for any other special analysis
- Configurable downlink modulation waveforms
- Multi-threaded, making use of multi-core processors to speed up analysis

Setting Up Fast Sequence Tuning

It is very easy to configure PXI 3000 Series Modular RF instruments to perform the simultaneous transmitter and receive calibration sweep using the Fast Sequence Tuning measurement suite (FST)

Configure the Sweep Parameter

- TX and RX Power level list (dBm)

A user can program the power levels the device transmits and the power levels the device receives in the form of two power arrays in dBm. These arrays are common to each frequency used in the sweep.

In the above example,

The Tx Power Array should be {16 , 27, 26, 24, 22, 19, 16, -40}

The Rx Power Array should be {-74, -68, -26, -26, -26, -26, -26, -26}

- TX and RX Frequency list (Hz)

A user can define the transmitter and receiver frequency channels in Hz.

In the above example,

The Tx Frequency array should be
{1924.2e6,1927.6e6,1931.2e6,1934.6e6,1938.2e6,1941.6e6

,1945.2e6,1948.6e6,1952.2e6,1955.6e6,1959.2e6,1962.6e6

,1966.2e6,1969.6e6,1973.2e6,1976.6e6}

The Rx Frequency array should be
{ 2114.2e6,2117.6e6,2121.2e6,2124.6e6,2128.2e6,2131.6e6

,2135.2e6,2138.6e6,2142.2e6,2145.6e6,2149.2e6,2152.6e6

,2156.2e6,2159.6e6,2163.2e6,2166.6e6}

When using the FST measurement suite, the frequencies can be in any band supported by the hardware. For example, instead of calibrating one GSM band at a time, you can combine all four bands in a single calibration sweep.

- **Step length (seconds)**

The device transmits each power level in the sequence for a predefined period of time. This is called the step length.

In GSM, the step length is normally a time slot, 577 μs.

In WCDMA, the step length is normally 20 ms although some devices can support shorter step lengths.

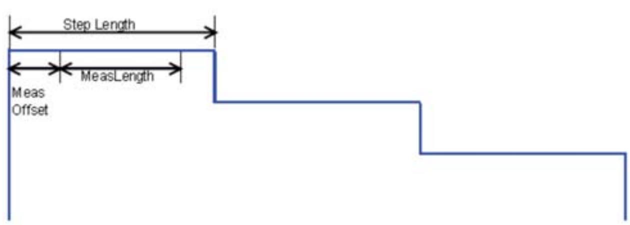
PXI 3000 Series modules can switch frequency in as short as 250 μs (Please refer to datasheets), there is no fixed upper limit for the step length (subject to PC and HW memory).

- **Measurement length (seconds)**
- **Measurement offset (seconds)**

FST allows you to define the portion of each step to measure by setting measurement length and measurement offset parameters. This permits measurements to exclude transient or unstable behavior from the mobile phone following a level or frequency change.

As shown in the below diagram, the measurement offset parameter marks the beginning of the valid period to measure.

The measurement length parameter defines the measurement period in each step.



In practical measurements, a typical step length might be 20 ms of which the mid 10 ms is used for measurement. The actual values used will be dependent upon the design and stability of the mobile phone.

- **Modulation source**
- **Aiq file name**

Different devices require different stimulus signals for receiver calibration. FST allows the user to select the signal source to be either CW (carrier wave) and ARB (arbitrary modulation). If ARB is selected, then, the user must also select which waveform file to play. The appropriate ARB file can either be selected from one supplied with the GSM/EDGE or UMTS UL measurement suite installer or created using IQCreator®. However, note that, if IQCreator® is used, the 3020 option 100 must be enabled for GSM/Edge and option 100 & 102 enabled for UMTS or cdma2k.

- **Enable measurements**

FST can perform the following analysis on the device's transmit signal:

- **Power**
- **Phase**
- **Frequency error**
- **I&Q data**

For any other key performance indicators, such as EVM, the user can get the I&Q data of each step and post process the data using either one of Aeroflex's analysis libraries or their own.

- **Sampling frequency (Hz)**

This parameter is used to configure the PXI 303x digitizers sampling rate. As a rule of thumb, the user must set the sampling frequency to be twice the signal bandwidth or higher.

Modulation	Recommended Sampling Frequency
WCDMA	3.84 MHz * 2 = 7.68 MHz (power/modAcc measurement only)
GSM	254 kHz * 4 = 1.08 MHz (power/modAcc measurement only)

- **Trigger source**
- **Trigger level (dBm)**
- **Trigger time out (milliseconds)**

FST allows the user to configure the instrument to synchronize a device in several ways.

A commonly used approach to synchronize the test instrument to the mobile device is to arm the instrument to trigger the sweep from receipt of the first mobile transmitter output using trigger source set to burst and an appropriate trigger level setting.

Untriggered (free running) sweeps may also be executed with the trigger source set to immediate.

For more options such as trigger polarity, trigger delay and trigger internal/external source, please refer to the FST Help file.

- **Headroom level (dBm)**

Complex modulated signals such as WCDMA have high peak to average (PAR) power ratios or crest factors. To avoid the signal peaks overdriving the digitizer hardware resulting in measurement errors, the user should let FST measurement know the expected crest factor. This is then used to adjust the 3030 digitizer RF input level setting accordingly.

- **System loss compensation**

Test system path losses will exist between the mobile antenna connection and the signal generator output and signal analyzer input.

FST allows the following two ways to adjust for these losses:

1) Input/output offset (dBm)

If path losses are frequency independent and symmetrical then a single global value can be set and used to compensate all measurements and hardware settings.

2) Get input/output loss handler

For any frequency dependent losses, such as Aeroflex PXI 3061 RF combiners, you can install Get input/output loss handler. For specific usage information, please refer to Help file and example codes.

Making Measurements

- **Measurement start**

After configuring the FST measurement, use measurement start to perform measurements.

- **Measurement WaitComplete**

Use WaitComplete to halt the program until the measurement is completed. Instead of polling for completion, this call does not waste any CPU power. In a multi-threaded test application, this period of time can be utilized to perform other tasks, such as adjustment value calculation.

Getting Results

After the measurement is completed, you can get ResultsSets containing all enabled results of the sweep, including:

- Power
- Phase
- Frequency error
- I&Q data

Assigning RF Hardware to FST

- **Assign Digitizer**
- **Assign SigGen**

RF hardware must be acquired before performing the measurements.

There are two ways of assigning RF hardware resources to the FST measurement library:

1) by resource string

In this mode, FST automatically boots and controls the RF hardware. This provides the easiest way of controlling PXI modules.

2) by instance handle

For DUTs supporting multiple modulation schemes, the same test application may need to use other measurement libraries in addition to FST. In this situation, the user can pass in the instance handles of the digitizer and sigGen to FST. This reduces the time to switch between different modulation modes by avoiding repeatedly booting the modules.

To get the best performance, it is recommended that the FST measurements are configured upfront and the hardware resources are assigned to FST as and when needed, e.g. prior to starting the measurement. After the measurements using FST are completed, make sure the digitizer and SigGen are unassigned from FST before performing other operations on the same hardware.

Conclusion

Simultaneous transmitter and receiver calibration is an innovative approach to reduce test time. The Fast Sequence Tuning (FST) measurement suite is a new and easy tool to help test engineers in their strive to speed up calibration. Based on the list mode of Aeroflex PXI RF analyzers and signal generators, FST can significantly reduce your alignment time.

Further Information

Fast Sequence Tune (FST) is a measurement library .dll supplied as standard with each of the following Aeroflex measurement suites: GSM/EDGE (2.5.0 onwards), UMTS UL (3.3.0 onwards) and cdma2k/1xEV-DO (2.5.0 onwards). Users may update to the latest version that includes FST by visiting the Aeroflex web site www.aeroflex.com\pxi.

For new customers, FST is supplied as a component within options 100 (GSM/EDGE), 101 (UMTS UL), 102 (cdma2k/1xEV-DO) for the Aeroflex 3030 Series of RF Digitizers.

All options may be retrofitted to an existing 3030 Series digitizer.

Contact Aeroflex sales for further information about ordering.

For further information regarding advanced features and other applications, please refer to

- Help file
- Example code: www.aeroflex.com/pxi_sample_code

Example Code for a Typical TXRX Sweep

The following section is a code example simulating a TXRX sweep with eight power levels repeated on sixteen frequencies.

```
#include <stdio.h>
#include <stdlib.h>
#include <afMeasureFSTDll.h>

/* ---------------------------------------- */
/* -- FST Instance handle               -- */
/* ---------------------------------------- */
afMeasureFST_Instance_t        fst;

#define DIGITIZER_LO_RESOURCE "PXI2::15::INSTR"
#define DIGITIZER_RF_RESOURCE "PXI2::14::INSTR"
#define SIGGEN_LO_RESOURCE     "PXI2::11::INSTR"
#define SIGGEN_RF_RESOURCE     "PXI2::10::INSTR"
static void AssignHardware(void) ;
static void MeasurementSetup(void) ;
static void PerformMeasurement(void) ;
static void MeasurementResults(void) ;
static long errFST(long status);
static void CleanUpAndExit();

/* -------------------------------------------------------------------------- */
int main()
{
    /* -- Create instance of required object -- */
    if (afMeasureFSTDll_CreateObject(&fst) < 0)
    {
        printf ( "Failed to create FST object\n");
        CleanUpAndExit();
    }

    AssignHardware() ;
    MeasurementSetup() ;
    PerformMeasurement() ;
    MeasurementResults() ;

    /*- Destroy Object -*/
    afMeasureFSTDll_DestroyObject(fst);
    fst = 0;

    CleanUpAndExit();
}

/* -------------------------------------------------------------------------- */
static void AssignHardware()
{
    long  status = 0;
    printf("\n======== Assign Hardware ========\n") ;

    /*- Assign Siggen and Digitizer to the FST measurement library -*/
    /* When assigning siggen and digitizer, the user must consider the 10MHz reference
     * frequency standard setup.  If a 3011 is associated with siggen (or digitizer), the user
     * must assign siggen (or digitizer) first, setting freqStandardIsInternal to AF_TRUE.*/
    status  = errFST(afMeasureFSTDll_AssignDigitizerByResource (fst, DIGITIZER_LO_RESOURCE,
                                                                DIGITIZER_RF_RESOURCE, AF_TRUE));
    status |= errFST(afMeasureFSTDll_AssignSigGenByResource (fst, SIGGEN_LO_RESOURCE,
                                                             SIGGEN_RF_RESOURCE, AF_FALSE));
    if (status < 0)
    {
        printf("Failed to assign hardware\n");
        CleanUpAndExit();
    }
}
```

```c
/* ---------------------------------------------------------------------------------- */
static void MeasurementSetup()
{
    double digFreq[]    = {1924.2e6, 1927.6e6, 1931.2e6, 1934.6e6,
                           1938.2e6, 1941.6e6, 1945.2e6, 1948.6e6,
                           1952.2e6, 1955.6e6, 1959.2e6, 1962.6e6,
                           1966.2e6, 1969.6e6, 1973.2e6, 1976.6e6};
    double siggenFreq[] = {1924.2e6, 1927.6e6, 1931.2e6, 1934.6e6,
                           1938.2e6, 1941.6e6, 1945.2e6, 1948.6e6,
                           1952.2e6, 1955.6e6, 1959.2e6, 1962.6e6,
                           1966.2e6, 1969.6e6, 1973.2e6, 1976.6e6};

    float  digLevel[]    = {-74, -68, -26, -26, -26, -26, -26, -26};
    float  siggenLevel[] = {-74, -68, -26, -26, -26, -26, -26, -26};

    long   timeout = 10000; /* milliseconds */
    double steplen = 10e-3; /* seconds */
    double measlen = 4e-3;  /* seconds */
    double measoffset = 1e-3;  /* seconds */
    long   status = 0;
    printf("\n======== Measurement Setup ========\n") ;

    /*--------------------------*/
    /*- Configure sweep        -*/
    /*--------------------------*/
    /* Configure sweep frequencies and levels */
    status |= errFST(afMeasureFSTDll_TxFrequencies_Set(fst,
                       digFreq,sizeof (digFreq)/sizeof(digFreq[0])));
    status |= errFST(afMeasureFSTDll_TxLevels_Set      (fst,
                       digLevel,sizeof(digLevel)/sizeof(digLevel[0])));

    status |= errFST(afMeasureFSTDll_RxFrequencies_Set(fst,
                       siggenFreq,sizeof (siggenFreq)/sizeof(siggenFreq[0])));
    status |= errFST(afMeasureFSTDll_RxLevels_Set      (fst,
                       siggenLevel,sizeof(siggenLevel)/sizeof(siggenLevel[0])));

    /*- configure steps - */
    status |= errFST(afMeasureFSTDll_StepLength_Set    (fst, steplen));
    status |= errFST(afMeasureFSTDll_MeasLength_Set    (fst, measlen));
    status |= errFST(afMeasureFSTDll_MeasOffset_Set    (fst, measoffset));

    /*- Configure signal generation -*/
    status |= errFST(afMeasureFSTDll_ModulationSource_Set (fst,afMeasureFST_msCW));
    status |= errFST(afMeasureFSTDll_TrainRfLevels_Set     (fst, AF_TRUE));

    /* ------------------------------------- */
    /* -- Configure measurement parameters -- */
    /* ------------------------------------- */
    status |= errFST(afMeasureFSTDll_Measurement_Enabled_Set (fst, afMeasureFST_miPower, AF_TRUE));
    status |= errFST(afMeasureFSTDll_Measurement_Enabled_Set (fst, afMeasureFST_miPhase, AF_TRUE));
    status |= errFST(afMeasureFSTDll_Phase_AxisUnits_Set      (fst, afMeasure_puRadians));
    status |= errFST(afMeasureFSTDll_SamplingFrequency_Set    (fst, 15.36e6));
    status |= errFST(afMeasureFSTDll_HeadroomLevel_Set        (fst, 0.0));
    /* Configure trigger mode */
    status |= errFST(afMeasureFSTDll_Trigger_Source_Set  (fst, afMeasureFST_tsImmediate));
    status |= errFST(afMeasureFSTDll_Trigger_Timeout_Set (fst, timeout));

    /* ------------------------------------ */
    /* -- Two ways of handling loss       -- */
    /* -- 1. Fixed offsets (dB)          -- */
    /* -- 2. Callbacks (see help)        -- */
    /* ------------------------------------ */
    status |= errFST(afMeasureFSTDll_InputOffset_Set  (fst, 0.0f));  //Tx - digitizer
    status |= errFST(afMeasureFSTDll_OutputOffset_Set (fst, 0.0f)); //Rx - siggen
}

/* ------------------------------------------------------------------------------------------------ */
static void PerformMeasurement()
{
    long  status = 0;
    printf("\n======== Perform Measurement ========\n") ;

    /*--------------------*/
    /*- Start measurement -*/
    /*--------------------*/
    if (errFST(afMeasureFSTDll_Measurement_Start(fst)) < 0)
    {
        printf("Failed to start measurement\n");
        CleanUpAndExit();
    }
```

```
    /* wait for measurement to complete */
    status = errFST(afMeasureFSTDll_Measurement_WaitComplete (fst, -1));
}

/* ------------------------------------------------------------------------------------------- */
static void MeasurementResults()
{
    long   numOfSteps = 0;
    int    i = 0;
    long   status = 0;
    int    index = 0;
    long   len = 0;
    float *powerArray = 0;
    float *phaseArray = 0;
    afMeasureFST_ResultsSet_t result;

    printf("\n======== Measurement Results ========\n") ;

    /*--------------------*/
    /*- Get result set -*/
    /*--------------------*/
    if (errFST(afMeasureFSTDll_Measurement_ResultsSet_Get(fst, &result)) < 0)
    {
        printf("Failed to get result set\n");
        CleanUpAndExit();
    }

    for (;;)
    {
        /* ------------------------------ */
        /* -- Get measurement results -- */
        /* ------------------------------ */
        status |= errFST(afMeasureFSTDll_NumberOfSteps_Get (result, &numOfSteps));
        if (numOfSteps == 0 || status < 0)
            break;

        /* Get power results */
        powerArray = (float*) malloc (2 * numOfSteps * sizeof(float));
        if (powerArray == NULL)
        {
            printf("Failed to allocated memory\n");
            break;
        }
        status |= errFST(afMeasureFSTDll_Power_Get (result, &powerArray[0], numOfSteps));
        if (status < 0)
            break;

        /* Get phase results */
        phaseArray = &powerArray[numOfSteps];
        status |= errFST(afMeasureFSTDll_Phase_Get (result, phaseArray, numOfSteps));
        if (status < 0)
            break;

        /* print results to screen */
        status |= errFST(afMeasureFSTDll_TxLevels_GetLength(fst, &len));
        printf ("Index    Power(dB)    Phase(rad)\n");
        for (i = 0 ;i < numOfSteps; ++i)
        {
            if (len > 0 && i % len == 0)
            {
                printf("\nFrequency : %d\n", index);
                index ++;
            }
            printf("%d\t%8.2f\t%8.2f\n", i, powerArray[i], phaseArray[i]);
        }
        break;
    }

    if (powerArray != 0)
    {
        free (powerArray);
        powerArray = 0;
    }

    /* User must destroy any result objects manually */
    afMeasureFSTDll_DestroyObject(result);
}
```

```
/*-----------------------------------------------------------*/
/*- Display error and warning messages -*/
/*-----------------------------------------------------------*/
long errFST(long statusCode)
{
    if (statusCode)
    {
        char errMsg[255];
        long pLength;

        afMeasureFSTDll_GetErrorMsgLength(fst, statusCode, &pLength);
        afMeasureFSTDll_GetErrorMsg(fst, statusCode, errMsg, pLength);

        if(statusCode < 0)
            printf( "Error: %s\n", errMsg);
        else if(statusCode > 0)
            printf( "Warning: %s\n",errMsg);
    }
    return statusCode;
}

void CleanUpAndExit()
{
    /*-----------------------------------------------------------*/
    /*- Destroy FST measurement library instance -*/
    /*-----------------------------------------------------------*/
    if (fst != 0)
        afMeasureFSTDll_DestroyObject(fst);

    printf("Press ENTER to exit");
    getchar();
    exit(0);

}
```

CHINA Beijing
Tel: [+86] (10) 6539 1166
Fax: [+86] (10) 6539 1778

CHINA Shanghai
Tel: [+86] (21) 5109 5128
Fax: [+86] (21) 6457 7668

FINLAND
Tel: [+358] (9) 2709 5541
Fax: [+358] (9) 804 2441

FRANCE
Tel: [+33] 1 60 79 96 00
Fax: [+33] 1 60 77 69 22

GERMANY
Tel: [+49] 8131 2926-0
Fax: [+49] 8131 2926-130

HONG KONG
Tel: [+852] 2832 7988
Fax: [+852] 2834 5364

INDIA
Tel: [+91] (0) 80 4115 4501
Fax: [+91] (0) 80 4115 4502

JAPAN
Tel: [+81] 3 3500 5591
Fax: [+81] 3 3500 5592

KOREA
Tel: [+82] (2) 3424 2719
Fax: [+82] (2) 3424 8620

SCANDINAVIA
Tel: [+45] 9614 0045
Fax: [+45] 9614 0047

SPAIN
Tel: [+34] (91) 640 11 34
Fax: [+34] (91) 640 06 40

UK Cambridge
Tel: [+44] (0) 1763 262277
Fax: [+44] (0) 1763 285353

UK Stevenage
Tel: [+44] (0) 1438 742200
Fax: [+44] (0) 1438 727601
Freephone: 0800 282388

USA
Tel: [+1] (316) 522 4981
Fax: [+1] (316) 522 1360
Toll Free: 800 835 2352

AEROFLEX
A passion for performance.

www.aeroflex.com
info-test@aeroflex.com

Our passion for performance is defined by three attributes represented by these three icons: solution-minded, performance-driven and customer-focused.

Part No. 46891/988, Issue 1, 11/09