MASTER THESIS

TITLE: Digital predistortion by using GPIB-controlled instrumentation

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Juan Murillo Espinar

DIRECTOR: Gabriel Montoro López

DATE: June, 5th 2007

Title: Digital predistortion by using GPIB-controlled instrumentation

Author: Juan Murillo Espinar

Director: Gabriel Montoro López

Date: June, 5th 2007

Overview

A digital predistortion using real equipment, holding it with new software, is here presented.

The Vector Signal Analyzer (VSA) –by Agilent– is a new software which allows to the user complete functionalities for the study of real signals. This software consists basically in a Spectrum Analyzer, regarding its performance, but increasing functionalities and commodity.

With the VSA software one can have a complete control of an overall communication system, taking advantage of its capacity to share data with other applications. By this way, data information of the signal received on a Spectrum Analyzer can be obtained and studied. In this study is showed how the VSA allows taking data in different modes. Besides, using the COM API language, it is possible to control the VSA with other softwares. Using this performance, and combining it with GPIB (General Purpose Interface Bus), the complete management of the whole system is achieved from Matlab. The GPIB allows interconnecting both Signal Generator and Spectrum Analyzer devices with the PC.

Once this connection is reached and all the parameters are well specified, the main goal of this Master Thesis is to turn the VSA software transparent to the user. The final scenario is to send a signal from Matlab and taking it –when the signal has passed across a power amplifier (PA)– again from Matlab (by means of the VSA software, but making it invisible to the user).

In order to prove the correct performance of the system implementation, a digital predistortion is employed. Thus, once the digital predistortion is done, the performance of the overall system should increase. This is because the nonlinearities due to the PA should be solved.

Herein, problems solved, VSA parameters adjustments, Matlab code program and main results of the digital predistortion, having in mind its future implementation in a FPGA, are presented.

ÍNDEX

INTRODUCTION

This Master Thesis presents the results of a digital predistortion. Nevertheless, it is important to say that the main effort is achieve to realize this predistortion by using a GPIB-controlled instrumentation and a new software called VSA (Vector Signal Analyzer) from Agilent.

The GPIB (General Purpose Interface Bus) is a short range digital data bus which allows to connect hardware devices, as Signals Generators and/or Spectrums Analyzers, to some PC in order to control them remotely. Concretely, the SCPI (Standard Commands for Programmable Instruments) commands are used to manage the hardware from the PC.

Once the connection is accomplished, the VSA software is employed to obtain the information data that before it was able to be seen on a Spectrum Analyzer. The VSA software provides the traditional spectrum displays and measurements of a typical Spectrum Analyzer, but with some advantages:

- A part of the standard information, it has several new options, measurements and displays. For instance, a lot of actual and newer modulation formats (like EDGE, MSK, M-QAM, $^{\pi}/_{4}$ DQPSK …), such as spread spectrums or multicarrier modulations (OFDM), can be selected on the VSA software. Later, this point is exposed widely.

- Maybe the most important issue and advantage of the VSA software, although the first one is so outstanding, is the possibility to share data with other softwares, like Microsoft Excel or Matlab. It allows to have an absolute control of the overall communication system because the received signal could be completely taken in. It has to be mentioned that VSA software permits to get data in different modes (the demodulated data, the received symbols and the IQ information once the signal passes across the shaping filter or when it does not pass across…).

- Another significant aspect, and perhaps not so important, is the possibility to work with the PC instead of the hardware device. With the VSA, up to 9 different graphs could be seen at the same time on the PC screen, allowing a better control of the signal. A part of this, to work with the PC improves the commodity.

The VSA software has other key option. It can be managed from other applications or softwares by means of its COM API (Component Object Model Application Programming Interface) language. The main softwares that could control the VSA are ADS (Agilent Design System) and Matlab. The last one is used on this work.

One of the main goals of this Master Thesis is to research about the new software bought by the EPSC (Escola Politècnica Superior de Castelldefels), the VSA software. Because of the darkness of the software help and the enormous quantity of functions and parameters bad exposed, this process consumed a lot of time. In fact, the presence in a one-day course of this software, guided by Agilent workers, was required.

When the software is controlled, the goal is to reach that the VSA turns transparent to the user. It means that it must be completely controlled from Matlab. In order to verify the correct performance, a digital predistortion for linearise a power amplifier (PA) is implemented.

Nowadays, when the main requirements in modern communication systems –overcoat in mobile communications systems– are high data transfer rates and a long time of battery life, the predistortion is continuously under test.

In order to achieve high data transfer rates, modern multilevel and multicarrier modulations are currently used. What this implies is the presence of high PAPR (Peak to Average Power Ratios). And, thus, if a linear amplification wants to be achieving, the work point should be moved far of the compression point. It is well known that as much as is moved the work point away from the compression point, the battery life will be decreased considerably.

The main objective of this Master Thesis is achieved. A whole control of a complete system from Matlab, making the VSA invisible and transparent to the user, is made successfully. In order to realize the digital predistortion, a signal was created from Matlab and sent to a Signal Generator by GPIB commands. Then, once the signal pass across the PA, it is obtained by the VSA software and the data is collected again to Matlab software, with which is possible to compare the signal sent with the signal received and make the digital predistortion.

Once the main PA characteristics and nonlinear effects are exposed, some of principal nonlinearities compensation techniques are listed and digital predistortion is widely studied. Afterwards, hardware devices and GPIB connection employed in this work, such as VSA software, are here presented. In relation to the VSA software, all the most important concepts and the problems solved when this work was advancing, and the COM API way to interconnect VSA to Matlab software and the GPIB/SCPI commands used, are explained. Finally, the whole system is described and the digital predistortion results are shown and justified for its implementation in a FPGA.

CHAPTER 1. PA nonlinearities

If the PA input power is small, the PA performance is linear. In this case, the PA is working in its linear zone and, at the output, only the frequency components of the input appear. On the other hand, when the input power is high, the PA is working in its nonlinear zone.

The effects that produce the PA nonlinearity over modulated signals are mainly two. They are called “in-band effects” and “out-of-band effects”. The first one, the “in-band effects”, produces a constellation distortion and consequently, a worse BER (Bit Error Rate) value. The second group, the “out-of-band effects”, produces a spectrum widening and so, a higher ACPR (Adjacent Channel Power Ratio) value. These two impacts are, obviously, important disadvantages.

Being the PA input signal represented in (1.1), at the PA nonlinear output will appear spurious at other frequencies. These are grouped in harmonic zones that should be separated by filtering. The output would be the equation represented in (1.2).

$$x(t) = a(t) \cdot \cos\{w_c t + \varphi(t)\} \quad \textbf{(1.1)}$$

$$y(t) = \mathrm{A} \cdot |a(t)| \cdot \cos\{w_c t + \varphi(t) + \Phi|a(t)|\} + \sum_{n=2}^{\infty} B_n \cdot |a(t)| \cdot \cos\{n w_c t + \varphi(t) + \Psi_{\mathrm{n}}|a(t)|\} \quad \textbf{(1.2)}$$

Where *n* is the harmonic zone, being the first zone the more conflictive in communication systems. The third order harmonics are in this first zone. These are not easy to remove due to their proximity to the signal, whereas the other harmonics are easy to remove using of a filter. According to (1.2), the signal at the first harmonic zone showed in (1.3).

$$y(t) = \mathrm{A} \cdot |a(t)| \cdot \cos\{w_c t + \varphi(t) + \Phi|a(t)|\} \quad \textbf{(1.3)}$$

1.1. PA identification

There are different representative parameters to identify a PA, like the gain, the frequency range or the low noise figure, among others. Two of these key parameters are the *k* order interception point and the compression point.

1.1.1. *K* order interception point

One of the main identification parameters of a PA is the *k* order interception point (IP_k). It is ever specified on the datasheet. Usually the third order (IP_3) is specified because, as it has been seen before, it is the more conflictive. The input IP_3 (IIP_3) is the input signal level, when the fundamental tone intensity at the input coincides with the third harmonic level. It is impossible to reach to the

IIP_3 because it will be ever a higher power value than the saturation power level[1]. Thus, in order to determine the IP_3, an extrapolation of the linear zone on both curves has be done. It is showed on Fig. 1.1. The harmonics are one of the principal reasons of the ACPR existence.

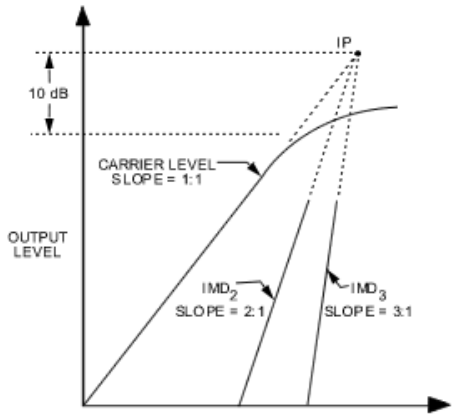


Fig. 1.1 Interception point [1]

1.1.2. Compression point

Other important parameter on the PA identification is the compression point (usually specified the 1dB compression point). As it has been explained, the PAs have a saturation power. This is known as gain compression effect. The input power when the fundamental output power is 1dB besides the ideal (linear) level is called “1dB input compression point”. The gain compression effect is one of the principal reasons of the constellation distortion at multilevel modulations, because different amplitudes values are amplified with different gains.

The best way to study the gain compression effect consists in draw the AM/AM static distortion curve. The functions $A|a(t)|$ and $\Phi|\mathrm{a}(t)|$ in (1.3) are known as AM/AM (amplitude modulation/amplitude modulation) and AM/PM (amplitude modulation/phase modulation), respectively. The first one represents the input amplitude modulation versus the output amplitude modulation and the second represents the input amplitude modulation versus the phase error between the input and output modulation. As it can be seen at Fig. 1.2, if the PA is working in its nonlinear zone, it means close to the “1dB compression point”, the signal will be degraded. In order to specify clearly when the PA is working, the back-off parameter is defined.

[1] The saturation power level is the maximum power that can provide the PA. It means that, although the input power level would increases, the PA could not provide higher power.

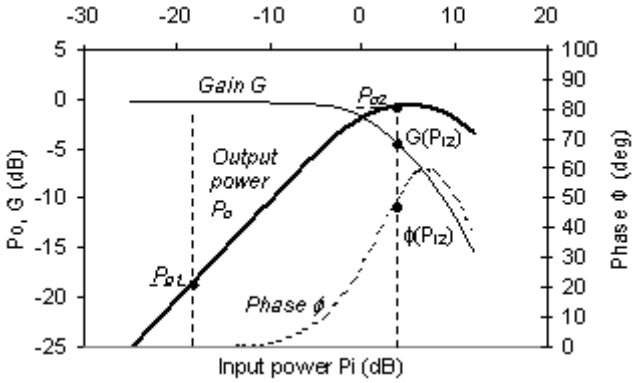


Fig. 1.2 Gain compression curve, 1 dB compression point (AM/AM curve), AM/PM curve [2]

It has to be in minding that, although the AM/AM curve is usually characterized with just one line, in the amplitude modulation scenario it should be represented by points. Because the signal sent is by points, the received signal has a relation with its symbol rate. So, in order to characterize the AM/AM curve, the input symbol module versus its output symbol module has to be located on the graph with one point. Thus one point is represented for each symbol sent. And, on the other hand, to characterize the AM/PM curve, the input symbol module versus the phase error between input and output symbols is represented. An example of this is showed at Fig. 1.3, and an AM/AM and AM/PM curves results are at Fig. 1.4.

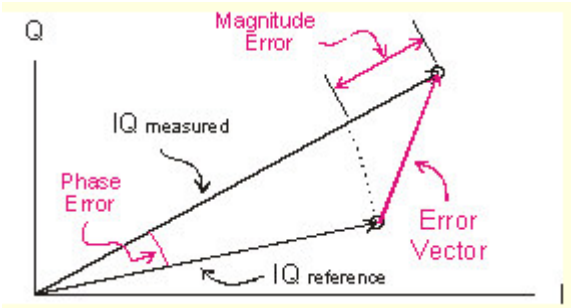


Fig. 1.3 AM/AM and AM/PM characterization points

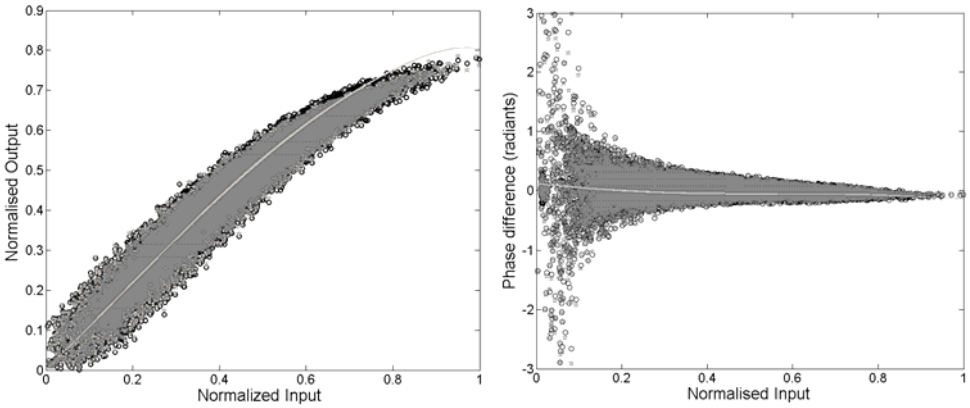


Fig. 1.4 (a) AM/AM curve, (b) AM/PM curve, both represented by points

It is interesting to observe the blurring effects on the AM/AM and AM/PM curves. This is consequence of the memory effects of the PA under test. These effects will be explained afterwards on chapter 2. The digital predistortion, a part of solving the nonlinearity of that curves, it will be able also to solve these memory effects considerably.

1.1.3. Back-off

From the AM/AM curve, the back-off value is defined in order to know where the PA is working. The back-off is the value[2] that relates the PA work point with the PA 1dB compression point [3]. As in the IP_3 and in the 1dB compression point, the back-off can be defined at the input (input back-off, IBO) and at the output (output back-off, OBO).

Thus, if it is said that it is working at IBO or OBO of 0dB, it means that it is working exactly over the compression point, where the signal will be degraded significantly. When the IBO value (or ever equivalent to OBO) is increased, the work point is moving away the compression point and so is working at a more linear zone and, consequently, with less distortion. The expression that related the IBO with the input power is like in (1.4).

$$InputPower = IP_{1dB} - IBO \quad \textbf{(1.4)}$$

It seems clear that is interesting to work at the linear region in order to against the PA nonlinearity. Nevertheless, there are some important negative consequences. Maybe the main disadvantage is that at higher IBO values, where the PA performance is more linear, the energetic efficiency is highly poor, involving a reduction of battery life, really important in mobile devices.

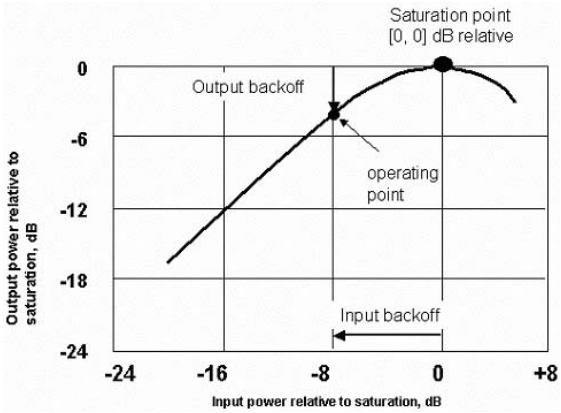


Fig. 1.5 IBO and OBO definition [4]

[2] Value in dB.

CHAPTER 2. DIFFERENT KINDS OF LINEARISER

Once the PA nonlinearities and its consequences on a signal are clarified, different possible solutions are listed here. Finally, the kind of lineariser used at this work is widely explained.

As it has been said before, the logic way to solve the nonlinearity is working at its linear zone and much far as it can be possible to the compression point. Taking into account that current personal standards use modern multilevel and multicarrier modulation formats -which presents high peak to average power ratio (PAPR) values-, higher values of IBO are necessary. However, this option is obviously bad if the energetic efficiency is considered. In order to have a good energetic efficiency and consequently, a larger battery life –aspect too important in mobile devices-, the work point should be close to the compression point. So, there are a trade off among the linearity and the battery life.

Having both linearity and battery life good enough, there are some ways to solve this trade off. The best option to have an efficient PA is the use of some technique that minimizes the PA nonlinearities. Here are listed the most common ones and their main characteristics are mentioned, although some of them shouldn't be used to compensate PA nonlinearities.

2.1. Nonlinearities compensation techniques

The main nonlinearities compensation techniques are listed here. But, as it will be explained later, some of these compensation techniques should not be used with the objective of linearise power amplifiers. Nevertheless, they can be used for other applications.

2.1.1. FeedForward

The FeedForward scheme can be seen at Fig. 2.1. The basic concept that the FeedForward technique follows is the distortion cancellation by means of two loops.

The main idea is that if the linear signal is subtracted to the signal plus the distortion, the final signal that will appear after this subtract is only the distortion. Then, once the distortion is identified, it will be also subtracted to the signal plus the distortion, obtaining so the final signal amplified without distortion. Maybe this idea seems too difficult, but if the different signals are studied at Fig. 2.1, the concept should be clarified.

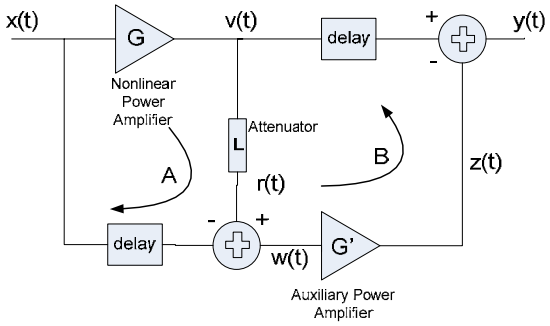


Fig. 2.1 FeedForward scheme

The A loop is called “the signal cancellation loop”, and the B loop is called “the distortion cancellation loop”. Other important issues are that: (1) the gain value *G* is equal than the *G'* value and (2) the attenuator gain is $L = \frac{1}{G}$.

The signal *v(t)* is just the input signal amplified by the nonlinear power amplifier. So, *v(t)* could be wrote as (2.1), that represents the input signal multiplied by the gain value plus the distortion.

$$v(t) = G \cdot x(t) + d(t) \tag{2.1}$$

where *d(t)* is the distortion.

Then, if *v(t)* passes across the attenuator, the *r(t)* signal is like in (2.2).

$$r(t) = \frac{v(t)}{L} = \frac{G \cdot x(t) + d(t)}{L} = \frac{G}{L} \cdot x(t) + \frac{d(t)}{L} = x(t) + \frac{d(t)}{L} \tag{2.2}$$

Because as it has been said before, *G* is equal than *L* and so $\frac{G}{L} = 1$. If *r(t)* is subtracted to *x(t)*, the result should be like (2.3).

$$w(t) = r(t) - x(t) = x(t) + \frac{d(t)}{L} - x(t) = \frac{d(t)}{L} \tag{2.3}$$

At this moment the *w(t)* signal is just the distortion provided by the nonlinear power amplifier and the signal is cancelled. Because of this, the loop is called “the signal cancellation loop”.

Afterwards, the *w(t)* signal passes across the auxiliary power amplifier with an equal gain than the first one, giving as a result the *z(t)* signal.

$$z(t) = G' \cdot w(t) = \frac{G'}{L} \cdot d(t) = d(t) \tag{2.4}$$

Now *z(t)* is just the distortion introduced by the nonlinear PA. So, the only thing that should be done now is to subtract *v(t)* in order to delete completely the distortion.

$$y(t) = v(t) - z(t) = G \cdot x(t) + d(t) - d(t) = G \cdot x(t) \tag{2.5}$$

Finally, the linear amplification without distortion is the output after the signal passes across these two loops.

The advantages of the FeedForward are that the PA gain is not reducing and the correction is not based in pass effects, it is based in what is happening at the moment. The FeedForward configuration is unconditional stable. Nevertheless, there are some disadvantages, too. The first one is the complex system circuit, which is an open loop circuit. Thus, it does not compensate the variations in time and temperature. Other important disadvantage is that, at this model, the auxiliary power amplifier is linear, and this is too difficult to achieve when the model is carried to the reality.

2.1.2. Feedback

This method is maybe the most obvious and simple in order to reduce the amplifier distortion. It is based on the use of some kind of feedback.

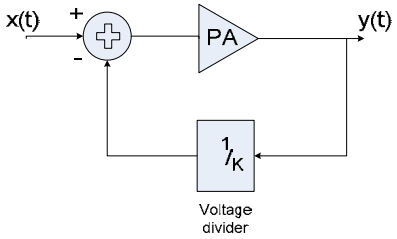


Fig. 2.2 Feedback scheme

The feedback is universally used for correct every type of error. However, in the case of radio frequency communication is not a good technique to reduce the nonlinearities, because it sacrifices the power amplification gain to improve the linearity. Moreover, the scheme could be unstable.

2.1.3. LINC (Linear Amplification with Nonlinear Components)

The LINC scheme is showed at Fig. 2.3. It is based in the law that it is ever possible to find two signals $v_1(t)$ and $v_2(t)$ with constant amplitude that accomplish the next relations.

Being $v(t)$ any band-pass signal

$$v(t) = A(t) \cdot \cos\left(w_0 \cdot t + \phi(t)\right) \quad \textbf{(2.6)}$$

It is always possible to find:

$$v(t) = v_1(t) + v_2(t) \quad \textbf{(2.7)}$$

$$\begin{cases} v_1(t) = A_{max} \cdot \cos(w_0 \cdot t + \phi(t) + \alpha(t)) \\ v_2(t) = A_{max} \cdot \cos(w_0 \cdot t + \phi(t) - \alpha(t)) \end{cases} \quad \textbf{(2.8) and (2.9)}$$

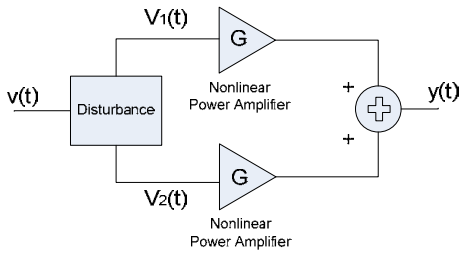


Fig. 2.3 LINC scheme

$$y(t) = G \cdot v_1(t) + G \cdot v_2(t) = G \cdot \big(v_1(t) + v_2(t)\big) \quad \textbf{(2.10)}$$

Following (2.7), (2.10) can be reduced in (2.11), where finally just the lineal amplification is obtained.

$$y(t) = G \cdot v(t) \quad \textbf{(2.11)}$$

The main drawback is that the disturbance of the signal is difficult and is not trivial. Thus it is usually implemented by DSP.

2.1.4. ACG (Automatic Control Gain)

This compensation technique is clearly inadequate to employ at this study. At the ACG technique, the variable gain should be inverse proportional to the *v(t)* power. The ACG method is typically used on radio links.

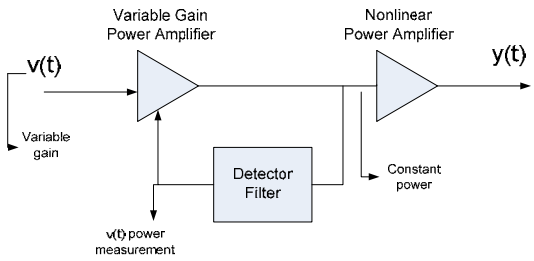


Fig. 2.4 ACG scheme

In this case, the main disadvantage is that it can not be implemented when the information is traveling on the amplitude.

2.1.5. Predistortion

The predistortion technique is the simplest idea to linearise a PA. Basically, it consists in create a previous distortion curve complementary to the power amplifier distortion (look at Fig. 1.2). So, if these two curves are situated in cascade, the final result is completely lineal. This is showed at Fig. 2.5.

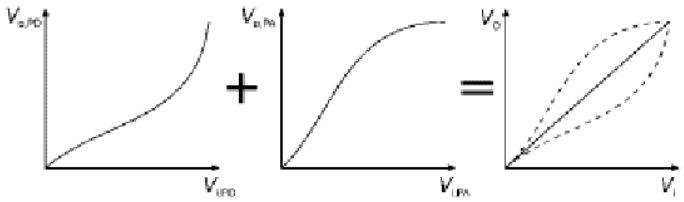


Fig. 2.5 Predistortion scheme

It has to be mentioned that it could exist analog predistortion or digital predistortion. Herein, the digital predistortion will be implemented. Digital predistortion can be applied at baseband or at intermediate frequency (IF), whereas the analog predistortion is applied at RF.

2.2. Digital predistortion

Predistortion [5] is basically a method by which one first stimulates a non-linear PA with baseband samples and, then, observe the results of that stimulus at the PA output. Then, the AM/AM and AM/PM effects of the PA are estimated. These estimated distortions are then removed from the PA by predistorting the input stimulus with their inverse equivalents.

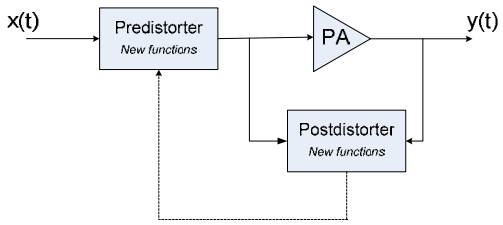


Fig. 2.6 Predistorter block scheme [5]

Therefore, the principal idea behind the concept of predistortion is the aim of introducing inverse nonlinearities that can compensate the AM/AM and AM/PM PA distortions.

Nowadays, the predistortion is an important issue to have into account owing to the modern multilevel modulation formats and multicarrier or spread spectrum techniques. For this, the digital predistortion has been object of multiple publications in the recent past years [6] [7]. In all these publications, a clear classification is applied in all the predistortion techniques used. The adaptive or non-adaptive predistortion and memory or memoryless effects are specified.

2.2.1. Adaptive / Non-adaptive predistortion

The difference among these two kinds of predistortion is very simple. The non-adaptive predistortion realizes the estimation of the predistorter function only one time and it is assumed that this curve will be ever valid for predistort the PA

input signal. The result of the non-adaptive predistortion is worse than if an adaptive predistortion is used.

On the other hand, the adaptive predistortion is the most suitable option to achieve a good estimation in all transmission time. The adaptive predistortion consists in obtaining output samples and estimating the predistorter function continuously. Therefore, if an adaptive predistortion is used, one can assure the correct and actual predistorter function for each moment, achieving a better linearization result than when a non-adaptive predistortion technique is implemented.

Inside the adaptive predistortion, a lot of different techniques are studied at papers. Herein, a non-adaptive predistortion and an adaptive predistortion with LMS (Least Mean Square) algorithm will be studied and the results will be compared.

2.2.2. Memory / Memoryless effects

Other important issues are the memory effects of the PA. Inside the different studies and publications related to digital predistortion, a clear differentiation is used: predistortion having into account the memory effects or without them (memoryless).

The memory effects are a consequence of the electrical and thermal dispersion effects [8]. However, there are other aspects related to the modulation formats or signal bandwidth that could be also relevant. If memory effects are presented, the PA output (amplitude and phase) not only depend on the instantaneous PA input, it depends also on their past values.

Thus, in the memoryless case, it is assumed that the AM/AM and AM/PM static curves will be always suitable for the same envelope value. So then, a table of predistorter gain values can be stored for every possible input envelope value. If this table is applied to the PA input, then it should cancel the undesired PA nonlinear response.

If modern communication systems are present, the digital predistortion based on memoryless models that only takes into account the AM/AM and AM/PM static curves does not achieve good results. It will be a right option with narrowband signals. With modern modulation schemes used nowadays as M-QAM or with modern techniques as WCDMA or OFDM, PA memory effects can not be ignored. When modulation bandwidth is relatively wide –more than 20 MHz– the PA starts to suffer from memory.

Against memoryless case, now for every particular value of input, there is more than one predistortion value that is needed to linearise the gain. The outcome that produces the memory effects are clearly seen on Fig. 1.4. As it can be seen at this figure, it produces blurring effects on the AM/AM and AM/PM curves.

Here is an example extracted from [9] where a 2.5 MHz bandwidth OFDM signal is measured by an Ericsson 45 W base station PA. At Fig. 2.7 and Fig. 2.8, the results for memory, memoryless and without predistortion are showed. It can be observed the difference between these measurements and how improve the results when the memory effects are taken into account.

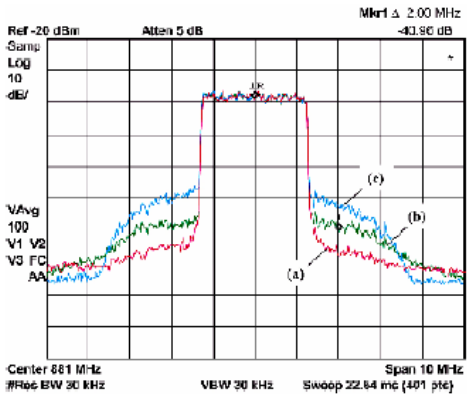


Fig. 2.7 (a) memory polynomial predistorter, (b) memoryless predistorter and (c) without predistortion

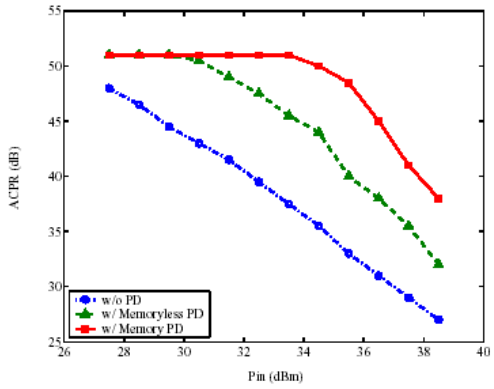


Fig. 2.8 ACPR for memoryless polynomial model, memory polynomial mode and without predistortion

2.2.2. Predistortion techniques employed

As it has been commented before, two kinds of predistortion will be compared.

- Non-adaptive predistortion
- Adaptive predistortion based on the LMS algorithm

The results of these two types of predistortion will be showed on chapter 6.

2.2.2.1 Non-adaptive predistortion

The non-adaptive predistortion is just the estimation of the static distortion curves and the later implementation of the predistorter curve.

It will be observed that, with this technique, the linearization will be achieved, but it will be the worse algorithm for the estimation of the predistorter curve. Then, adaptive predistortion algorithm is used in order to obtain a better result.

2.2.2.2 Adaptive predistortion – LMS algorithm

The Least Mean Square (LMS) algorithm, introduced by Widrow and Hoff at 1959 is an adaptive algorithm, which uses a gradient-based method of steepest decent. LMS algorithm uses the estimates of the gradient vector from the available data. LMS incorporates an iterative procedure that makes successive corrections to the weight vector in the direction of the negative of the gradient vector which eventually leads to the minimum mean square error. Compared to other algorithms, the LMS is relatively simple; it does not require correlation function calculation nor does it require matrix inversions.

How the LMS algorithm works is showed at Fig. 2.9 and at the equations (2.12) and (2.13) [10].

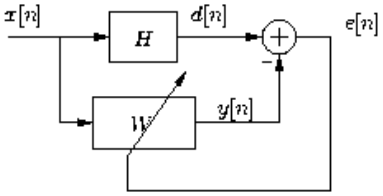


Fig. 2.9 LMS system identification block diagram

The adaptive filter W is adapted implementing the LMS, which is the most widely used adaptive filtering algorithm. First, the error signal *e* is computed as (2.12).

$$e = d - y \tag{2.12}$$

It measures the difference between the output of the adaptive filter and the output of the unknown system. On the basis of this measure, the adaptive filter will change its coefficients in an attempt to reduce the error. The coefficient update relation is a function of the error signal squares and is given by (2.13) [11].

$$h_{k+1} = h(k) - \mu \cdot \nabla_{\varepsilon} h(k) \tag{2.13}$$

In (2.13), *h* is the vector of filter parameters to be adapted $[h_1 \dots h_N]^T$, μ is a constant that determines the rate of adaptation, and $\nabla_{\varepsilon} h(k)$ is an estimation of the gradient of *h* with respect to the mean squared error, $\varepsilon = E[e^2]$. Equation (2.13) attempts to increment the filter parameter vector by small steps in the direction of decreasing mean squared error. Stochastic gradient adaptation proceeds by iterating (2.13) until the mean squared error is minimized.

Relating to this work, the LMS allows a better linearization. The predistorter function does not depend only to the actual values; it depends also on the past values. If *μ* parameter is a high value, the result depends in a greater way from the actual values. And on the other hand, if *μ* parameter is a low value, the results depends in a greater way from the past values.

2.2.2.3 Look-up-tables (LUTs)

In computer science, a Look-up-table (LUT) is a data structure, usually an array or associative array, used to replace a runtime computation with a simpler lookup operation [12]. The speed gain can be significant, since retrieving a value from memory is often faster than undergoing an expensive computation.

A classic example is a trigonometry table. Calculating the sine of a value every time such a sine is needed can be prohibitively slow in some applications. To avoid this, the application can take a few seconds when it first starts to pre-calculate the sine of a number of values, for example for each whole number of degrees. Later, when the program wants the sine of a value, it uses the lookup table to retrieve the sine of a nearby value from a memory address instead of calculating it using a mathematical formula.

There are two fundamental limitations on when it is possible to construct a lookup table for a problem. One is the amount of memory that is available; it is not possible to construct a lookup table larger than the space available for the table, although it is possible to construct disk-based lookup tables at the expense of lookup time. And the second restriction is the time required to initially compute the table values - although this does not need to be done often. If it requires prohibitive time, it may make the table inappropriate to be used.

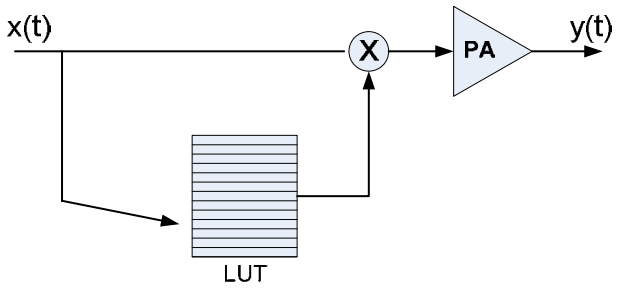


Fig. 2.10 Basic performance of a LUT system

CHAPTER 3. HARDWARE

3.1. Signal Generator Device -- E4433B Agilent

The Agilent E4433B RF signal generator [13] offers a wide range of digital modulation capabilities for research and development, manufacturing or troubleshooting applications. Providing a comprehensive feature set, it generates standard and custom digital modulation formats, filtering and burst shapes, as well as versatile analog modulation, with superior quality, reliability and worldwide support. It has the possibility to charge arbitrary signals from Matlab software –technique implemented here–.

Main features:

- 250 KHz to 4 GHz frequency range, with a resolution of 0.01 Hz
- RF modulation bandwidth up to 35 MHz
- -136 dBm to 7dBm power range, with a resolution of 0.02 dB
- Optional dual arbitrary waveform generator and/or real-time I/Q baseband generator
- 40 MHz sample rate and 14-bit I/Q resolution
- 1 Msample (4MB) memory for waveform playback
- 1 Msample (4MB) memory for waveform storage
- Custom digital modulation (>15 variations of FSK, MSK, PSK and QAM)
- AM, FM, phase modulation, pulse modulation and step/list sweep (frequency and power)
- Programming language: SCPI

Fig. 3.1 E4433B Agilent

This device is useful in this study because it is possible to download a waveform by using the GPIB bus and Matlab software. So, it can be controlled every time the signal. Finally, with the help of the VSA software (explained on chapter 4), it is possible to obtain the final data and send it to Matlab software in order to compare the signal sent versus the signal received, after this one has passed across the PA, for example.

3.2. Spectrum Analyzer Device -- E4407B Agilent

The HP Agilent E4407B ESA-E Series [14] is Agilent's mid-performance spectrum analyzer. The series sets the performance standards in measurement speed, dynamic range, accuracy, and resolving power for similarly priced products. Selection of one button measurement solutions combined with its easily navigable user interface and performance in speed allows the user to spend less time testing and more time designing, building, and troubleshooting components and products.

Main features:

- 9 KHz to 26.5 GHz frequency range
- Resolution bandwidth: 1 kHz to 5 MHz in a 1, 3, 10 sequence, and 5 MHz
- Phase noise: -90 dBc/Hz (10 kHz offset). 99dB third order dynamic range
- Overall amplitude accuracy: + or -(0.6dB + absolute frequency response)
- Absolute amplitude accuracy + or -1.1dB
- Measurement range: 50 ohms -120 dBm to +30 dBm
- Maximum safe input continuous power: +30dBm(1W)

Fig. 3.2 E4407B Agilent

3.3. PA Device – ZRL-2300 Minicircuits

The PA used in this work is the ZRL-2300 from Minicircuits. Here are listed its main characteristics[3].

- High IP3, +46dBm typ.
- Low noise figure, 2.5dB typ.
- Gain, ≈24dBm typ.
- Frequency range, 1.4 – 2.3GHz
- Applications: defense and satellite communications, PCS, UMTS, GSM, cellular, wireless data.

[3] The complete datasheet information is placed on the annex.

Fig. 3.3 ZRL-2300 Minicircuits PA

However, there is a problem using this PA for this study. Against the usual requirement of the commercial PAs, a high nonlinearity is here necessary in order to appreciate the improvements of the lineariser. It means that another worse PA should be used. Nevertheless, it is not possible because there is any else PA on the laboratory used for the development of this Master Thesis. But, actually, as it will be seen on chapter 6, the results are good enough and the predistortion effects can be appreciated correctly.

3.4. GPIB Bus – National Instruments

3.4.1. A brief history

The GPIB bus is a short range digital data bus implemented at 1965 by Hewlett-Packard (HP). Nevertheless, this first bus was called HP-IB (Hewlett-Packard Interface Bus). The goal to design the HP-IB was to connect the HP test and measures devices to some equipment in order to be able to program it, as for instance, a computer.

This standard was very useful and was quickly standardized by the IEEE (Institute of Electrical and Electronics Engineers) at 1975 and it was called IEEE-488 or GPIB (General Purpose Interface Bus). This last one is more widely used than HP-IB. Some of the principal reasons because this standard was quickly gained popularity are the high transfer rates -on the original standard was 1Mbps (later extended to 8Mbps)- and the number of devices that can be connected at same time -a maximum number of 15-.

Finally, it must be mentioned is that at 1990, the original standard was revised and, specifically, how the controllers and instruments communicate between them. The SCPI (Standard Commands for Programmable Instruments) commands were chosen, allowing a single and simple programming language that is used with any SCPI instrument.

In addition to the IEEE, others committees standardized the initial HP-IB. The ANSI (American National Standards Institute) standardized the HP-IB as ANSI Standard MC 1.1 and the IEC (International Electrotechnical Commission) has its IEC Publication 625-1.

Others GPIB revisions and improvements has been done. A complete time-line of the HP-IB/GPIB bus is showed at Fig. 3.4.

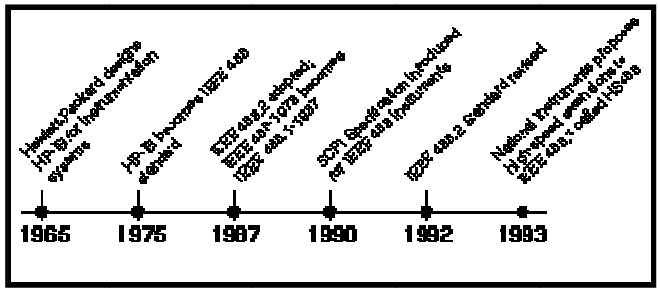


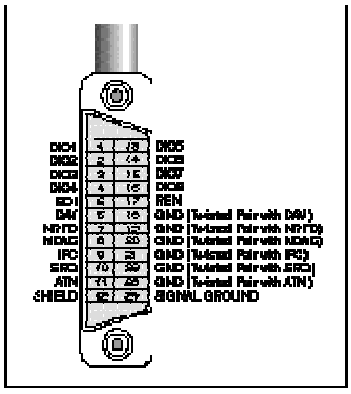
Fig. 3.4 HP-IB/GPIB bus time-line

3.4.2. GPIB Specifications

The GPIB devices communicate among them by means of two basic messages, the *Device-dependent messages*, often called data messages. These contain device-specific information, such as programming instructions, measurement results, data files, etc. And, on the other hand, the *Interface messages*, also called command messages, that contain message as initializing the bus, addressing devices, etc…

A part of this classification, the GPIB devices can be *Talkers*, which send data messages to one or more *Listeners*, which receive de data, and the *Controller* device, that manages the flow of information on the GPIB by sending commands to all devices.

DATA LINES	Pin No.
DIO1	1
DIO2	2
DIO3	3
DIO4	4
DIO5	13
DIO6	14
DIO7	15
DIO8	16



MANAGEMENT LINES	Pin No.
IFC	9
REN	17
ATN	11
SRQ	10
EOI	5

HANDSHAKE LINES	Pin No.
DAV	6
NRFD	7
NDAC	8

Fig. 3.5 GPIB connector

As it can be seen at Fig 3.5, the GPIB interface system consists on 16 signal lines and 8 ground-return or shield-drain lines. The 16 signal lines are grouped into 8 data lines, 3 handshake lines and 5 interface management lines.

- The 8 *Data lines*, called DIO1 to DIO8, carry both data and command messages. All commands and most data use the 7-bit ASCII or ISO code set, in which case the DIO8 is used for parity.

- The 3 *Handshake lines* control the transfer of message bytes between devices. It guarantees that the message sent is received without errors.

  - NRFD – *Not ready for data*. It indicates when a device is ready or not to receive a message byte.
  - NDAC – *Not data accepted*. It indicates when a device has or has not accepted a message byte.
  - DAV – *Data valid*. It indicates that the signals on the data lines are valid and are accepted by devices

- The 5 *management lines* manage the flow of information across the interface.
  - ATN – *Attention*. ATN=true implies that the data lines are sending commands, ATN=false implies that a Talker can send data messages.
  - IFC – *Interface Clear*. It initializes the bus.
  - REN – *Remote Enabled*. It places devices in remote or local program mode.
  - SRQ – *Service Request*. To request service from the Controller.
  - EOI – *End or Identify*. To mark the end of a message string or to identify their response in a parallel poll.

Other important specifications are that, nowadays, the GPIB bandwidth is 8Mbps, the maximum number of devices are 15 at the same time and the maximum cable length is 20 meters. A final history and specifications summary are shown on Table 3.1.

Table 3.1 GPIB history and specifications summary

Type	General Purpose Data Bus
Production history	
Designer	Hewlett-Packard
Designed	Late 1960s standardized in 1975
Manufacturer	Hewlett-Packard
Produced	1960s to present
Superseded by	VXI (in ATE) [15] (1990s)
Specifications	
External	Yes
Data signal	Parallel data bus with handshaking
Width	8 bits
Bandwidth	8 Mbps
Max devices	15
Protocol	Parallel
Cable	20 m
Pins	24 (8 data, 5 bus management, 3 handshake, 8 ground)
Connector	24-pin Amphenol-designed micro ribbon [16] [17]

Finally, three connection configurations can be implemented in order to connect the GPIB devices: the linear configuration, the star configuration or a combination of both.

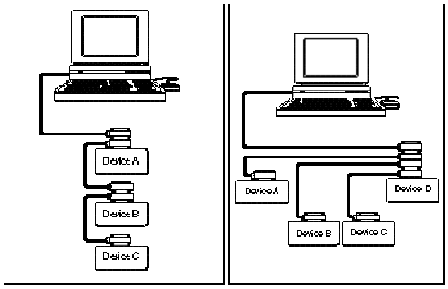


Fig. 3.6 a) Linear configuration, b) Star configuration

With the objective of connect the different devices to the PC, a GPIB card has been installed. The model card used is a National Instruments GPIB card (see Fig. 3.7).

Fig. 3.7 a) GPIB card for PC by National Instruments

3.4.3. Programming GPIB (SCPI)

The SCPI defines a standard set of commands to control programmable test and measurement devices in instrumentation systems. It specifies a command structure and syntax for programmable instruments control; for example, GPIB.

It was created at 1990; anyway the SCPI Consortium [18] continues adding commands and functionality to the SCPI standard. Before this date, each instrument manufacturer developed its own command sets for its programmable instruments. SCPI offers numerous advantages to the test engineer. For instance, it decreases development time and increases the readability of test programs and the ability to interchange instruments.

The SCPI devices uses a compose head system in order to provide a hierarchy structure to the commands. The command tree is a quite way to the implementation and documentation of the joint SCPI commands. At Fig. 3.8 the subsystem SENSe command tree is showed.

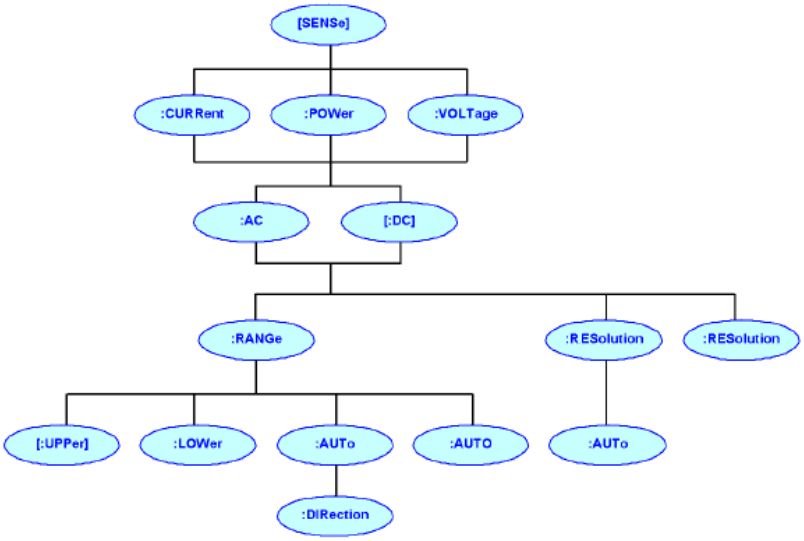


Fig. 3.8 SCPI command tree of the subsystem SENSe.

Each device and instrument has its own programming guide where the SCPI commands are specified. It is showed besides an example of SCPI command for the Agilent case. In order to select the center frequency, the programming guide gives details about the commands that must be used:

:FREQuency:CENTer
Supported All with Option 007
[:SOURce]:FREQuency:CENTer <num>[<freq suffix>]
[:SOURce]:FREQuency:CENTer?
This command sets the center frequency for a ramp sweep. The center frequency symmetrically divides the selected frequency span and is coupled to the start and stop frequency settings.

The rules used for the command sentence structure are the next ones:

- The brackets ([ ]) are optional parameters.
- The keys ({ }) specify a group of parameters that must be selected one of them.
- The vertical line (|) divide the different choices that could be chosen.
- A parameter between the symbols <> means that it must be provide its value or name.
- It has to be mentioned that in order to program the SCPI commands, two ways can be implemented. The long way; it means to copy all the line (the capital and the small letters), or the short way; it means to copy only the capital letters.
- Some instructions allow a question mark at the end of the line.

The GPIB/SCPI commands used in this work will be completely listed on the program code in the annexes.

3.4.4. GPIB used at this work

The GPIB controller has been used here, and which allows the devices connection, is the NI GPIB-USB-HS (National Instruments GPIB Controller for USB 2.0 High-Speed) [19]. Some of its principal characteristics are showed here, but the full datasheet of this GPIB controller also can be found at the annexes.

Main features:

- USB port: High Speed USB signaling, 480 Mbps
- IEEE 488 Compatibility: IEEE 488.1 and IEEE 488.2
- Maximum IEEE 488 Bus Transfer Rates:
    - IEEE 488 interlocked handshake: 1.8 Mbps
    - IEEE 488 non-interlocked handshake (HS 488): 7.2Mbps

Fig. 3.9 GPIB cables to interconnect the PC to devices and the devices between them

CHAPTER 4. SOFTWARE

4.1. VSA (Vector Signal Analyzer) software

One of the main objectives of that study is to know how this software works. It is a new software bought by the EPSC (Escola Politècnica Superior de Castelldefels). It is was necessary a lot of time because the documentation and help files of the VSA software are quite dark. With the realization of this work it is hoped that other future VSA user will be able to understand the main aspects of it and he could advance quickly on his work.

The VSA software is capable to offer a time-, frequency- and modulation-domain analysis providing measurements and displays as if a Spectrum Analyzer would be used. However, the 89600 Series VSA Software is more than a simple spectrum analyzer.

This software provides traditional spectrum displays and measurements, but today, spectrum analysis is not enough. New digital formats require new measurements. Familiar tools such as spectrum analyzers with demodulation may indicate that a problem exists, but they can not detect the cause of it. The VSA software provides the tools to identify the root cause of the problem and to analyze continually aspects as changing phase, magnitude and frequency.

The VSA also relies on a PC for its processing. So, in order to know the functionality of a complete system, the PC can help to the user. Data can come from several sources, including multiple supported hardware platforms, recorded files or, as in the case of this work, from the Matlab software by means of the GPIB bus and a signal generator. So, for this last case, the VSA software must be configured in order to communicate it with the signal generator, as it will be explained afterwards. Once the signal data is sent or played, in this work is used Matlab with the goal to take the final data to compare it with the data sent. All this Matlab connection is possible by the COM API, explained at section 4.2.

As a summary, it provides a Windows-based graphical user interface (GUI) for performing vector-signal analysis on data from Agilent Vector Signal Analyzers, from other hardware front ends such as the Agilent E4406, or from software products such as MATLAB, Advanced Design System, or Excel.

For instance, in relation to the modulation-domain analysis, in the 89600 Series VSA Software, some of the supported modulation formats which can be studied are the next ones[4]:

- BPSK, QPSK, 16-QAM, 64-QAM, 256-QAM, Pi/4 DQPDK, EDGE, CDMA, MSK…

[4] All the supported modulation formats are listed at the annex. Here is just presented some of the most important ones.

The display can be specified in different ways. Some of the information data that can be showed are the presents on Fig. 4.1[5]. The data could be showed by only one graph, two graphs, 2x2 grid… since a maximum of 9 graphs presented by 3x3 grid. At Fig. 4.1, it is illustrated an example of 2x2 grid.

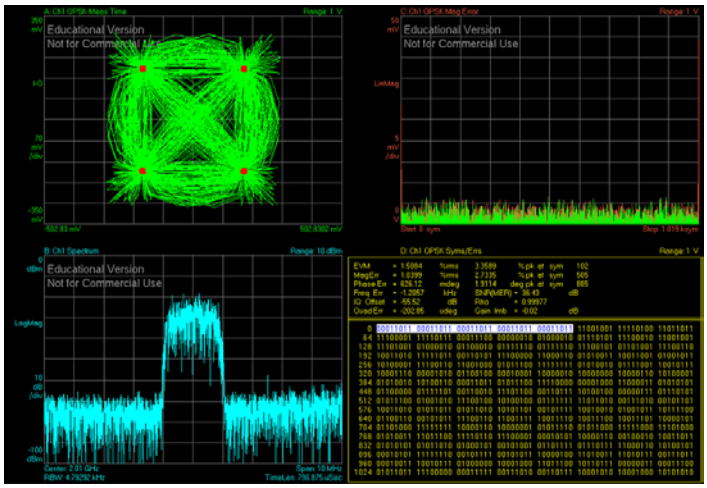


Fig. 4.1 VSA display example. QPSK modulation.

- *IQ Meas Time*.- For all demodulation formats except FSK, the analyzer's digital demodulator produces two signals: I/Q Measured and I/Q Reference. The IQ measured signal is the result of resampling the data to an integer number of points per symbol and applying carrier/symbol locking, IQ origin offset and amplitude droop compensation, system gain normalization, and filtering to the input signal. The filtering is user-selectable.

- *Spectrum*.- The Spectrum trace is the averaged or instantaneous of the spectrum display. If averaging is *OFF*, the averaged spectrum and the instantaneous spectrum displays are identical.

  The Spectrum trace has the following characteristics: (1) It is derived from pre-demodulated time data, which is 20% larger than the result length. (2) It is averaged if the averaging option is *ON* and the average type is RMS(Video), RMS(Video) Exponential, Time, Time Exponential or Continuous Peak Hold averaging.

- *IQ Mag Error*.- The IQ Mag Error and IQ Phase Error traces show the error between the I/Q measured and the I/Q reference signals. IQ Mag Error displays the magnitude error and IQ Phase Error displays the phase error. At previous Fig. 1.3, a graphical clarification is showed.

  If IQ Mag Error is selected, the analyzer compares the magnitude at the sampled symbol times of the I/Q measured signal with the magnitude of

[5] All the supported data formats are explained at the annex. Here is just presented the four showed at Figure 4.1.

the I/Q reference signal. Then, the analyzer displays the difference, in magnitude, between these two signals. If the normalization option is selected to *OFF*, the analyzer displays the instantaneous magnitude error –at this work this option is used in order to have the correct instantaneous value-, and if normalization is set to *ON*, the analyzer displays the magnitude error as a percentage.

- *Syms/Errs*.- Selecting the Syms/Errs, the trace data displays the symbol table. The symbol table shows the information error and the binary bits for each symbol. The first bit in the table corresponds to the first bit of the first symbol.

A part of the data information, a Y axis could be chosen between these ones:

- *Log Mag (dB)*; *Linear Mag*; *Real (I)*; *Imag (Q)*; *Wrap Phase*; *Unwrap Phase*; *I-Q*; *Constellation*; *Q-Eye*; *I-Eye*; *Trellis-Eye*; *Group Delay* and *Log Mag (lin)*.

As it is showed at Fig. 4.1, a lot of information could be represented simultaneously and at the same screen of a signal. As it has been said before, here is displayed 4 graphs, but it is possible to represent until 9 graphs at same time. Even it is able to select a concrete symbol at the syms/errs graph by a marker and connect it with the other graphs in order to know how or where is it.

There are a lot of options and possibilities to do with the VSA software but explain all of them is not a goal of this project. Nevertheless, some of the main and most practical utilities are listed here:

- *Player signal.-* The 89600-series VSA provides various recorded signals, analyzer setup files, and Signal Studio setup files. It is a suitable option if it is wanted to study some of these signals. For instance there are QPSK, O-QPSK, CDMA, 3GPP Down/Up, WiMAX 5MHz/7MHz, Zigbee, etc[6].

- *Record signal.-* The Vector Signal Analyzer application lets to record time data from the measurement hardware directly to the PC's disk drive. The data can be played back at a later time or import it into other applications. It could also create and play the user recordings. It is possible to specify the record length in time or in samples.

- *Marker.-* VSA includes several marker types and marker functions. The analyzer supports general trace marker functions and several specialized markers including; Band Power markers, Occupied Bandwidth (OBW) markers, Adjacent Channel Power (ACPR) markers, and Spectrogram markers. It is a good tool in order to know the relation between symbols or values of the signal from different graphs, for example.

[6] All the pre-recorded signals are listed at the annex with a little explanation. Here is just presented some ones.

- *Macro.-* It is a very useful option. Macros let to automate a series of manual operations into a single command. 89600-series products use VBScript[7] for their macro programming language. Macros can be used for:
    - One-button applications.
    - Automation of repetitive tasks.
    - Computation of measurement results that are beyond the scope of the basic 89600.

As a summary, the VSA software provides to the user a complete vision of all the main aspects in a complete communication system with the advantage to see different aspects of the same signal at same time.

Although the possibility to see until nine graphs of the same signal simultaneously, there is another advantage more relevant. Furthermore, the values of the signal that would receive the receptor can be taken to software. It means that the numerical values of all the graphs, that could represent the VSA, can be studied in software like Microsoft Excel or Matlab. For instance, the symbols demodulated, the values in quadrature and phase (I and Q), the magnitude or phase errors, the spectrum values, etc… can be taken to Matlab and be evaluated, modified and/or studied how the user wants. There are several ways to share data between the 89600 analyzer and other applications or programs.

This functionality is used in order to study the complete system. Without the VSA software, only the spectrum, the IQ draw and aspects as the EVM could be seen in a Spectrum Analyzer. Now, with the VSA software, the complete signal values can be studied. Actually, as it has been commented before, this is the way followed by this work to analyze the overall complete system.

The VSA can be controlled using menus and dialog boxes in the window, or controlling the application via the COM API application. Herein, the COM API is used and is explained at section 4.2.

[7] VBScript is a scripting language that is a subset of the Visual Basic programming language. VBScript programs are easy to create. With 89600-series products, it can be created a VBScript program by recording mouse and keyboard operations or it can be written (or edited) VBScript programs with the macro editor that comes with 89600-series software.

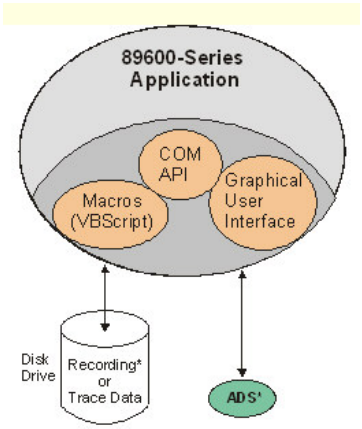


Fig. 4.2 VSA performance and relation scheme

4.2. COM API Matlab to VSA

The VSA software provides an Application Programming Interface to its Component Object Model, or COM API.

This COM API provides software engineers with pre-built objects, methods, properties, and constants to create applications that can use both the Vector Signal Analyzer and the Spectrum Analyzer applications. Together, these APIs expose all of the measurement, computational, and display features of the instrument, making them accessible to C++ or Visual Basic programs[8].

COM is an architecture and supporting infrastructure for building, using, and evolving software robustly. The model goes beyond ordinary object oriented programming in that it describes standard ways to define and create new interfaces. The COM standard is a programming model that describes how to connect objects and construct new interfaces.

Herein the COM API will be used to program and control the VSA software from Matlab. Thus, taking advantage of that, the vector signal generator can be also controlled from Matlab –by GPIB commands (SCPI language)-. A complete management of the overall system is finally achieved.

As example of the COM API language some instructions are showed below[9]. Firstly, the VSA 89600 object must be created from Matlab in order to have the total control of the VSA software with the next instruction.

```
hVSA = actxserver ('AgtVsaVector.Application');
```

[8] The provided APIs are officially supported from the following programming environments:
- Microsoft Visual Basic Version 5.0 or later, Enterprise and Professional editions.
- Microsoft Visual C++ 5.0 or later in container applications that support Component Object Model automation.

[9] The complete VSA object tree instructions are listed on the annexes.

Once this object is generated, it is possible to control the VSA options from Matlab. For instance, in order to select the option *Digital Demodulation* on the VSA, showed at Fig. 4.3, the next instructions must be specified on Matlab.

Furthermore, on Fig. 4.4, the object results in Matlab are represented.

```
hVSA = actxserver ('AgtVsaVector.Application');
hMeasurement = get(hVSA,'Measurement');
set(hMeasurement,'DemodConfig',2);
hDemod = get(hMeasurement,'DigDemod');
```

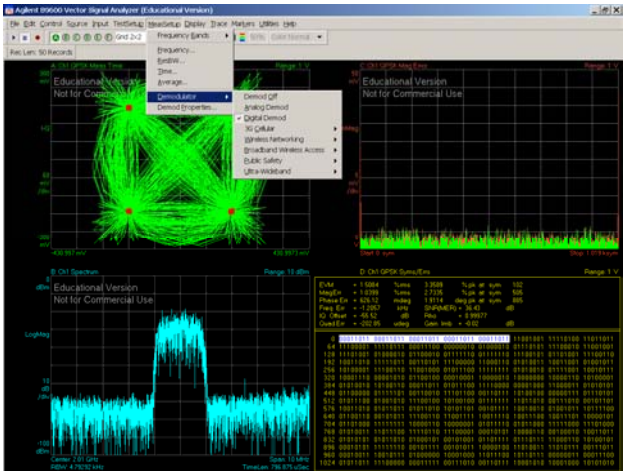


Fig. 4.3 Unfolded measurement and demodulation menu

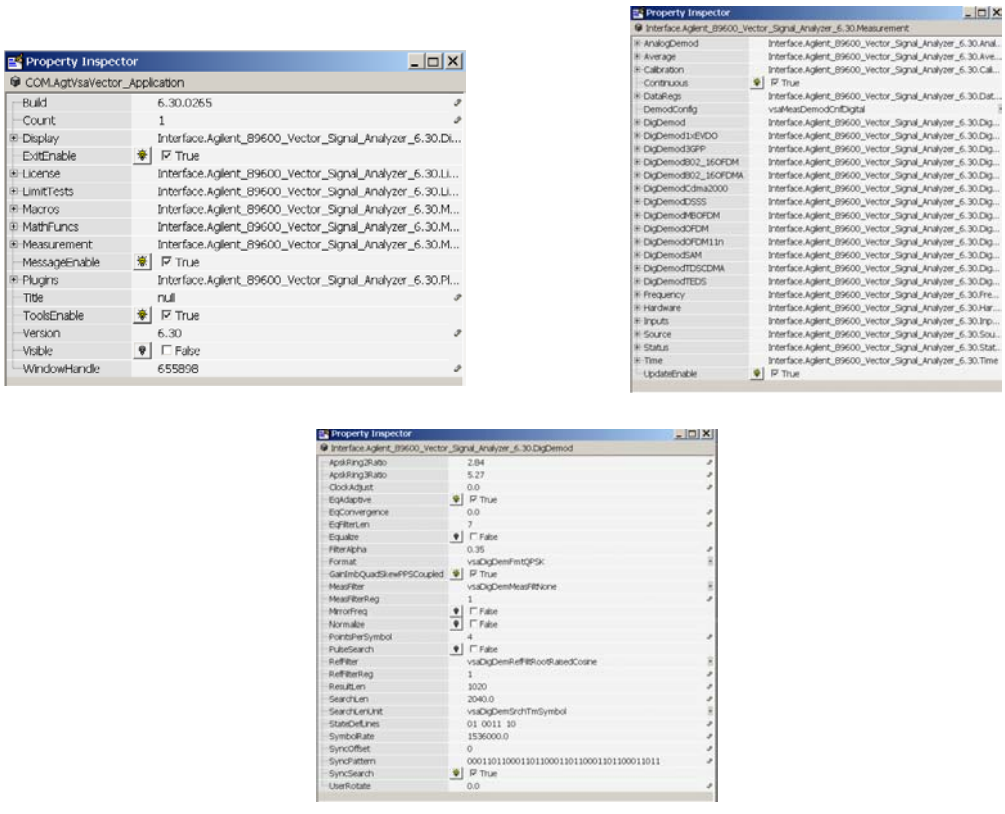


Fig. 4.4 (a) VSA, (b) Measurements and (c) DigDemod objects

CHAPTER 5. SYSTEM IMPLEMENTATION

5.1. Overall system

Then, the goal is to linearise a previous identified PA. It has to be mentioned that until nowadays, only the AM/AM curve has been linearised because of the impossibility to identify the phase values. Now, with the VSA software the phase can be also linearised. As it has been said before, this software can share data with other applications. So, if the phase information is taken from VSA, a predistorted phase signal could be implemented joint the typical predistorted amplitude signal.

The overall system scheme is showed at Fig. 5.1. The PC or workstation –with Matlab and VSA software previously installed– is connected to the signal generator by means of a GPIB bus. And then, it is connected also to the spectrum analyzer. Thus, three devices are used (workstation, signal generator and spectrum analyzer) among the 15 possible GPIB devices connected.

Then, the signal generator RF output is connected to the PA under test in port, and the PA under test output port is connected to the Spectrum analyzer RF in. Now, the total control of the complete system is achieved from the workstation.

Matlab is used to discover, identify, modify and control all the equipment of the overall system and to control the VSA software by means of the COM API.

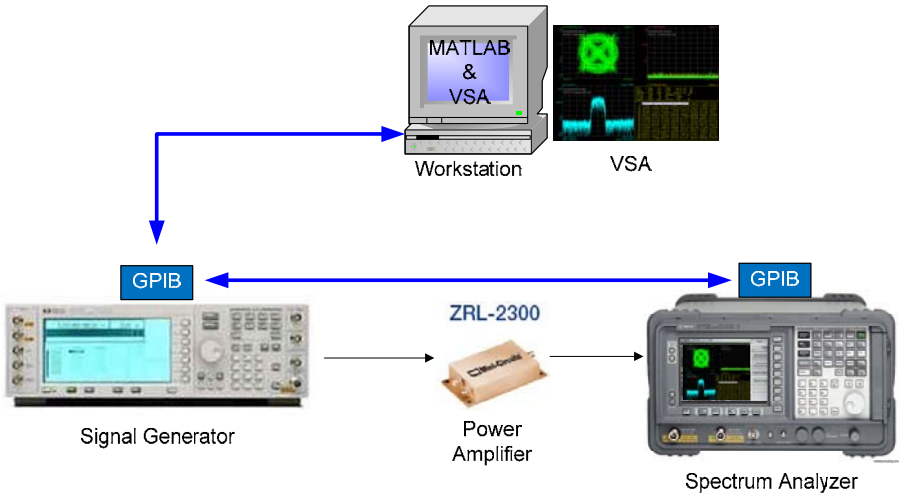


Fig. 5.1 Overall system scheme

The 10 MHz reference of the Signal Generator is used to adjust the clocks of all the systems jointly (VSA reference synchronized to Spectrum Analyzer reference, and also synchronized to the Signal Generator reference). The scheme followed for this purpose is showed at Fig. 5.2.

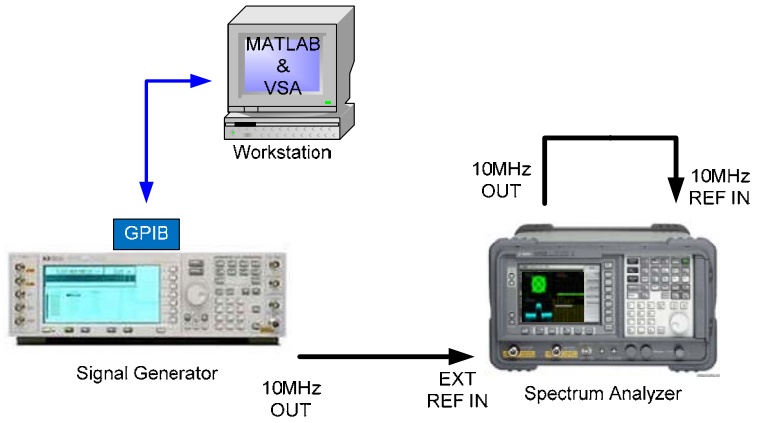


Fig. 5.2 Reference synchronization of hardware device and VSA software

5.1.1. GPIB commands specified

Although the complete programming code is placed on the annexes, the main GPIB commands required are here specified.

Each hardware device has an option to specify a GPIB address by a number. Once it is specified, when some instruction affect to these devices, it has to be referenced them by these addresses. At this work, the spectrum analyzer GPIB address is 18 and the signal generator GPIB address is 19.

Firstly, the devices joined by the GPIB cable should be identified. When the SCPI command *idn* (identify) is used, Matlab prompts the next identification.

```
% for the spectrum analyzer

g18=gpib('ni',0,18);
g18.InputBufferSize=50000;
fopen(g18)
fprintf(g18, '*IDN?');
idn = fscanf(g18)

% for the signal generator

g19=gpib('ni',0,19);
g19.InputBufferSize=50000;
fopen(g19)
fprintf(g19, '*IDN?');
idn = fscanf(g19)
```

```
idn =
Agilent technologies, E4448A, US43360350, A.08.09
(for the signal generator)

idn =
Agilent technologies, ESG-D4000B, GB40051154, B.03.86
(for the spectrum analyzer)
```

With the goal to linearise the PA, a signal should be sent from the signal generator. In order to have a whole control of the system, the signal is designed into Matlab.

Any type of signal that supports the VSA software –specified completely at the annexes– could be implemented on Matlab and sent it to the signal generator. Specifically, a 4-QAM signal is considered in this study. It is important to know all the characteristics of the signal to specify correctly the VSA parameters for the subsequent demodulation. The main characteristics of the 4QAM signal implemented to predistort the PA are showed at Table 5.1.

Table 5.1 4-QAM signal parameters

Symbols number	1000
Roll-off factor (α)	0.35
VSA Clock	6.144 MHz
Number points per symbol	4
Symbol rate (Clock VSA/points per symbol)	1.536 MHz
Carrier frequency	2.010 GHz
Power level	2 dBm
Filter order	80

These parameters have to be in mind because they will be used later to identify the VSA and the spectrum analyzer parameters as a receivers.

For instance, the frequency and power level SCPI commands for the spectrum analyzer are specified here. In this case, the center frequency is fixed to 2.01GHz and the span of the spectrum analyzer is fixed to 40MHz. It is important to observe that the specifications are sent as a string. It is possible to send the commands as bits, but it must be previously selected. This allows a highly speed at communication.

```
freq_cent=2.010;
cadena=[':FREQ:CENT ',num2str(freq_cent),' GHZ'];
fprintf(g18,cadena);

freq_span=40;
cadena=[':FREQ:SPAN ',num2str(freq_span),' MHZ'];
fprintf(g18,cadena)
```

Once the signal is made on Matlab, it is sent to the signal generator by means of GPIB commands. In order to send the signal, the load SCPI command is used, but it must be adapted to the dynamic range of the signal generator. At Fig. 5.3, a complete flow diagram of the GPIB-SCPI commands used to load the signal to the Agilent device is presented.

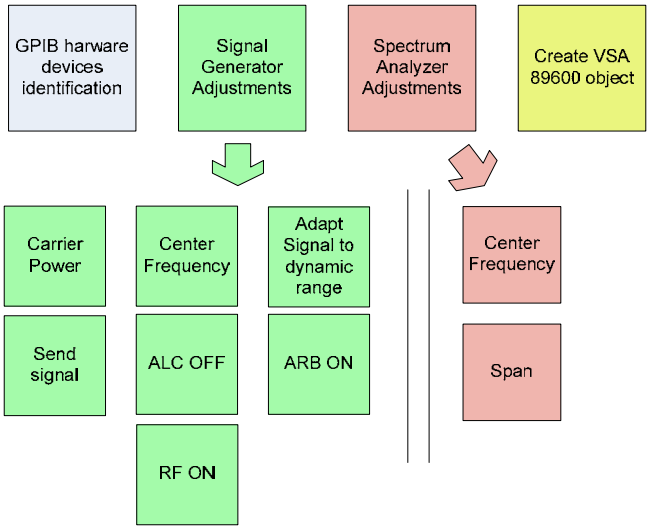


Fig. 5.3 Flow diagram of the GPIB-SCPI commands to send the signal

The first aspect that must be done, as it has been commented before, is the GPIB hardware devices identification in order to create an object in Matlab and so, to have the possibility to communicate with them. Later, the signal generator and spectrum analyzer adjustments must be done. At the first one, aspects as the carrier power and frequency of the signal are firstly adjusted. Then, the signal can be sent from Matlab by the *esg_darb* GPIB command. But before this, the signal created in Matlab should be adapted to the dynamic range of the signal generator in order to achieve a correct implementation.

The ARB (arbitrary waveform) is then activated. It allows to select the signal sent by the user. And finally, the RF output is changed to *ON* with the objective to allow to the Signal Generator launch the signal to the PA.

An important issue is the ALC (Automatic Level Control). When ALC is set to *ON*, the internal level detector watches the output level. So, it may not shift greatly from the set amplitude value. If the output level is greater than the specified level, the output amplifier's gain will be reduced, and if the output level is smaller, the output amplifier's gain will be increased. However, in some cases when the ALC is unable to maintain the output level, the unlevel message appears to notify the unleveled condition to the user. Thus, in order to allow the user change the amplitude from Matlab, the ALC should be OFF because if not, it will be never seen the correct amplitude value.

If the signal wants to be seen on the spectrum analyzer, it must be adjusted, at least, the center frequency and the span. On the contrary, if it will be seen on the PC by the VSA software, the VSA 89600 object must be created by means of the COM API commands –described before on section 4.2–.

The whole system is showed at Fig. 5.4. The Predistortion block is used just when the PA identification is finished[10]. On the VSA software, the signal parameters sent from Matlab should be specified for the correct demodulation.

[10] For more information, go to the section X.

Once the signal pass across the spectrum analyzer and the VSA object is created, all the “star” points at Fig. 5.4 could be studied from VSA software and, therefore, form Matlab. As it has been mentioned before, different information can be taken form the VSA software.

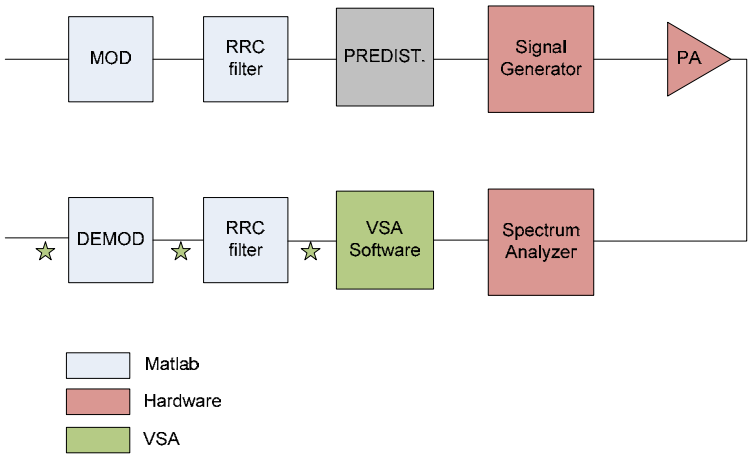


Fig. 5.4 Whole system (Matlab, hardware devices and VSA)

Although Fig. 5.4 shows a complete transmitter and receiver system –suitable if a correct demodulation of the signal sent wants to be realized–, this flow diagram is not correct in order to predistort the signal.

Here, a root raised cosine shaping filter is used in order to adapt the signal. As it has been widely explained on chapter 2.2, the PA input and PA output are compared to predistort the signal. Thus, the demodulated shaping filter must be deleted with the goal to obtain the data once it has passed across the PA.

How it must be done and all the defined VSA parameters values are clarified on next section.

5.1.2. VSA fixed and identified parameters

Once the main VSA characteristics are known –described on chapter 4–, the specific options and identified parameters used are here explained. It is widely specified the most important aspects considered and the mainly problems solved when the work was advancing.

5.1.2.1 Issues on parameters of the signal sent

The main parameters that must be selected on the VSA software to achieve a good demodulation data and to achieve the correct graphs are the parameters related to the signal information.

In concrete, firstly it has to be selected the modulation format –in this case a 4QAM modulation format is selected among the all possible formats allowed to

use with the VSA[11]-. Then, aspects as the number of samples per symbol and α parameter of the filter are chosen.

Finally, regarding to the signal information, another important issue is the symbol rate. Because the signal is made from Matlab and then taken out by the Agilent signal generator, the symbol rate is directly proportional to the reconstruction clock on the Agilent hardware device. On this device, the clock can be changed between a minimum value of 1 Hz and a maximum value of 40MHz. The clock selected is the defect value when the Agilent signal generator starts: 6.144MHz. Thus, in order to know the symbol rate of the signal sent to the PA, (5.1) should be followed.

$$Symbol\ rate = \frac{VSAclock}{number\ points\ per\ symbol} \quad \textbf{(5.1)}$$

5.1.2.2 Issues related to the measurement options

Different issues had to be into account. These ones are here listed.

- *Range.-* When the VSA is working, the range is quite important to obtained the correct values. If the input range is setting too low (more sensitive than necessary), the analyzer's ADC circuitry introduces distortion into the measurement. But if the input range is setting too high (less sensitive than necessary), there may be a loss of dynamic range due to additional noise. In some cases, the increase in the noise floor may obscure low-level frequency components.

  The right way to choose the correct range value is the “proof and error” method. When the range value is not the correct, a message will appear on the screen.

- *Measurement filter and reference filter.-* The Fig. 5.5 is used to try to clarify these parameters,.

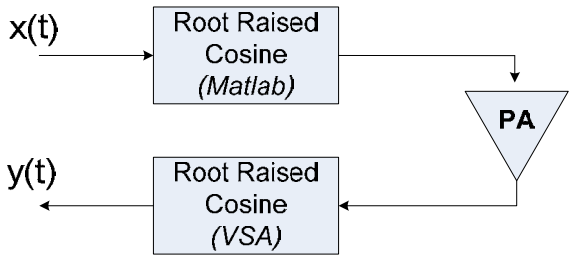


Fig. 5.5 Flow signal diagram

[11] All the modulation formats supported by the VSA software are showed on the annexes.

On Fig 5.4, the first RRC (Root Raised Cosine) filter is used by Matlab before the signal is sent, and the last one is used to demodulate the information.

The Measurement Filter is this last one. As it is mentioned on chapter 5.1.1, these two filters must be selected if the goal is to demodulate correctly the signal and close the whole loop. However, if the goal is to realize a predistortion, the last filter should not exist. So, in the case of the predistortion would be the objective, on the VSA software it has to be selected the option: *None* on the Measurement Filter.

And finally, for a suitable implementation, the Reference Filter has into account both shaping filters (the modulated and the demodulated RRC filters). It is the whole filter of the overall system. It means that, for instance, if both filters are RRC types, the overall filter will be a RC (Raised Cosine) filter. Or, if just the modulated shaping filter is RRC and the Measurement Filter is selected to *None* –as in the case of the digital predistortion implementation–, the Reference Filter must be a RRC filter.

- *Normalize.-* An important aspect that has to be into account to have a suitable result is to disconnect the Normalize option. As it has been said before on chapter 5.1.1, when the ALC issue was treated, if the Normalize option is switched on, aspects as the constellation points or the spectrum, will be normalized. It means that if the amplitude of the signal sent changes, this will not be appreciated by the VSA because it will normalize the signal values to standard values.

- *Length capture.-* This is a fundamental issue for synchronize the signal (explained afterwards). This user specification determines how many symbols should be demodulated for analysis.

  The length capture is the number of demodulated symbols –or equivalently any VSA data (IQ, EVM, and spectrum data …)– that will be captured and then, shared to others applications or software.

- *Search length.-* This is a fundamental issue for synchronize the signal (analyzed later). This user entry indicates to the demodulator how many symbols to search to find the user specified synchronization word.

  It is important to have into account that, with the goal to find the synchronization word, the search length should be at least the double of the length capture. If it is not in this way, maybe the word is outside the demodulated symbols fixed by the Length capture parameter.

- *Synchronism.-* Synchronism is used in order to achieve a compete coordination among the signal sent and the signal taken into Matlab. It is

reached with the Length capture and Search length parameters joint to a synchronization word. This synchronization word is defined by the user and is sent as a pilot message on the signal. Then, the VSA try to found this pilot message into the signal and, by this way, the signals sent and received are completely synchronized.

For instance, a 20 symbols pilot message (or synchronization word) is used. The symbols chosen to use in this study are:

[0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3]

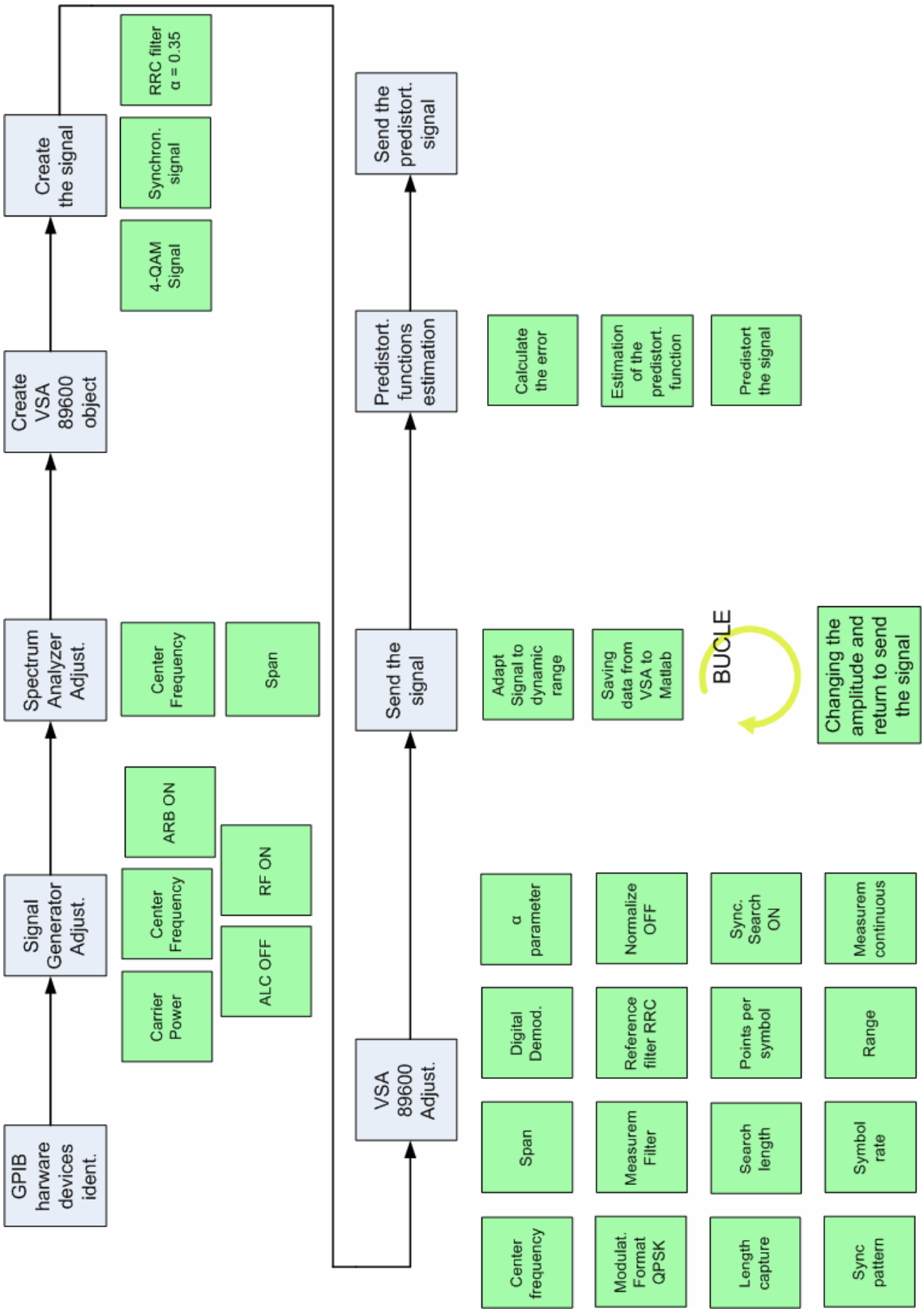
Their translation into bit information of a 4QAM signal is:

[00011011000110110001101100011011000110 11]

- *Measurement interrupted.-* Finally, the measurement should opt for an interrupted measurement, instead of continuous one.

  If it is selected the interrupted mode, the VSA makes just one actualization of the measurement and then it will hold this, although the Signal Generator continues sending the signal. Thus, when data will be taken into Matlab (with the COM API commands), all data types should be synchronized.

5.2. Program flow diagram



CHAPTER 6. FINAL RESULTS

6.1. Final whole system scenario

At Fig. 6.1 the final measurement scenario at laboratory 225 of the TSC (Teoria del Senyal i Comunicacions) is showed.

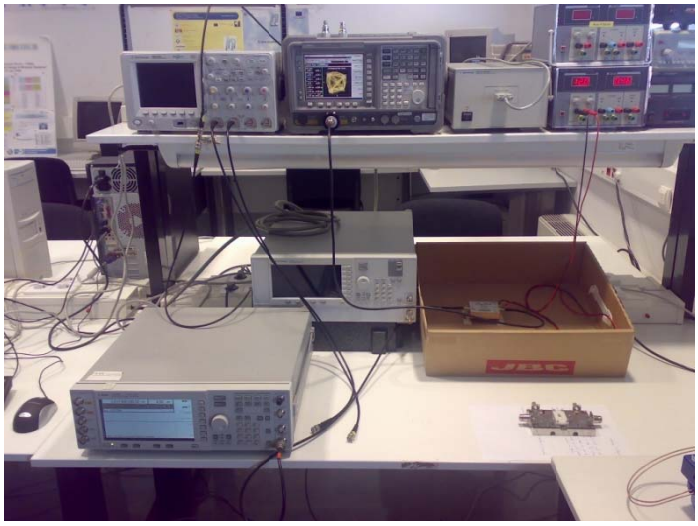


Fig. 6.1 Final whole system scenario

6.2. Non-Adaptive predistortion results (without LUTs)

6.2.1. PA identification

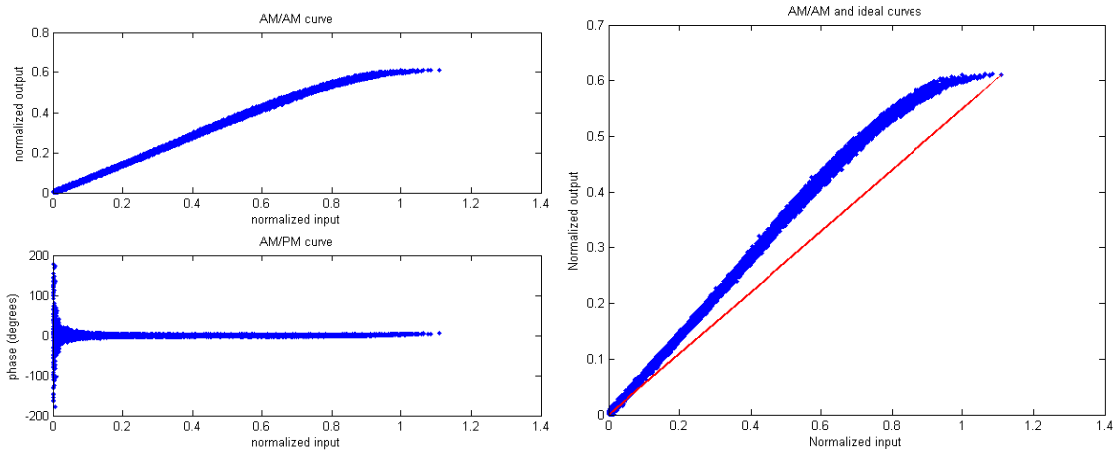


Fig. 6.2 (a) AM/AM and AM/PM curves, (b) AM/AM and ideal (linear) case

Firstly, the PA must be identified with its static distortion curves, represented on Fig. 6.2. As it has been mentioned before on chapter 3.3, this PA is extremely

good and so, the non-linearity and blurring effects could be scorned. In order to achieve a better result on this study, other worse PA should be used, but in fact, the results are good enough with the ZRL-2300 Minicircuits.

In this case, a ten times bucle is used, downing the amplitude of the signal sent with the objective of that all points between minimum and maximum values of the static curve would be excited. By this way the curve is continuous, as it can be seen on Fig. 6.2.

6.2.2. Predistortion curve identification

Then, using the program flow diagram showed before, the predistorter function is estimated.

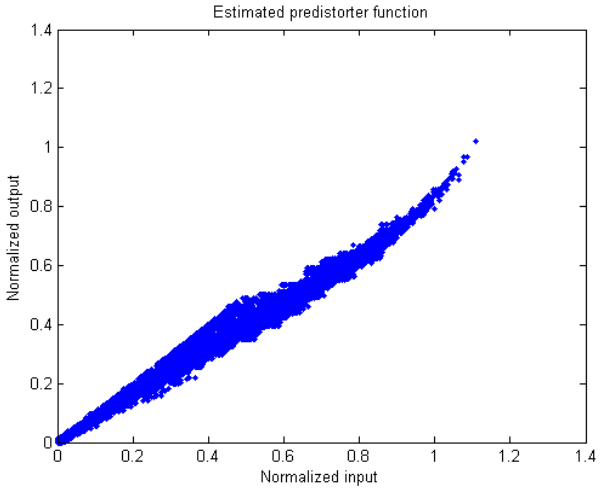


Fig. 6.3 Estimated predistorter function

6.2.3. Predistortion result

Once the estimated predistorter function is applied to the original signal, the final result is showed on Fig. 6.4. As it can be appreciated here, the result is linear and is following the ideal red line illustrated on previous Fig. 6.2 (b). It is clearly showed on Fig. 6.5, where all the curves are represented in just one graph.

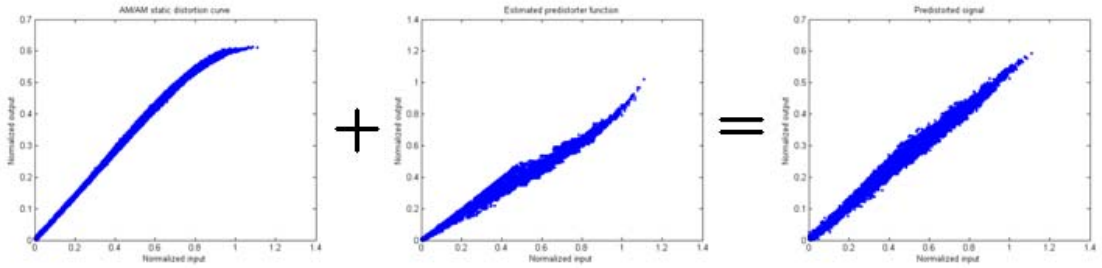


Fig. 6.4 Predistortion result

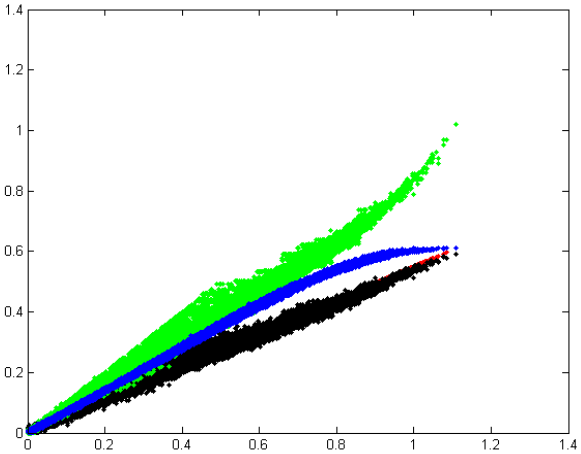


Fig. 6.5 All curves at one graph[12].

6.3. Non-Adaptive predistortion results (with LUTs)

With the goal to observe how the LUT works, a predistortion is realized also with this way. In this case just one iteration is done. The results are showed continuously.

6.3.1. PA identification

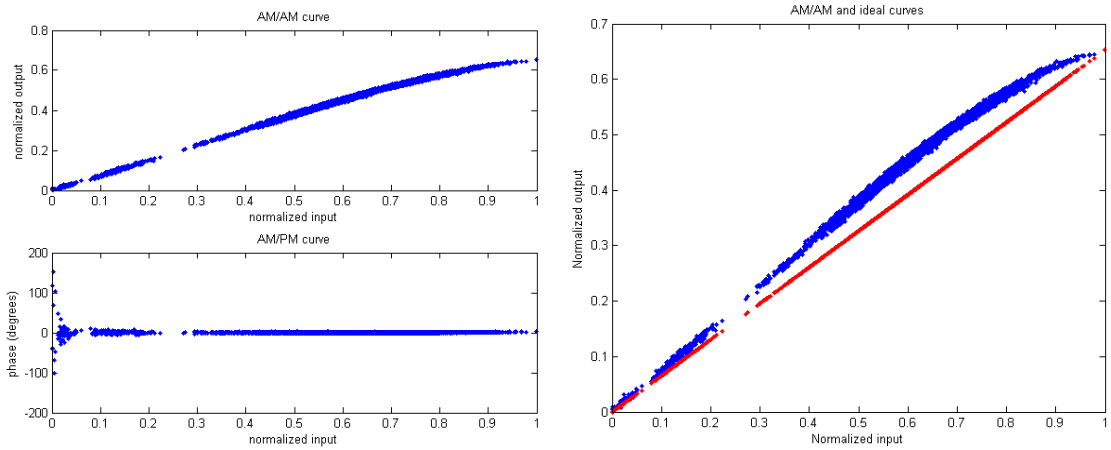


Fig. 6.6 (a) AM/AM and AM/PM curves, (b) AM/AM and ideal curve (LUT case)

[12] Notice how the predistortion result –dark curve– follow the same way that the ideal case –red curve–.

6.3.2. Predistortion curve identification (LUT values)

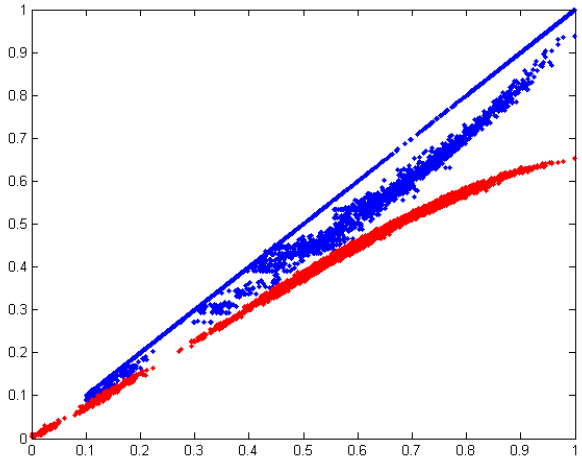


Fig. 6.7 Estimated predistorter function (LUT case)

Here, the upper blue line is due to the LUT initialization. Because of LUT is initialized by ones, the points of this LUT that are not excited by the PA will continue with this value. So, the blue points (estimated predistorter function) are all these LUT points that change its value. In Fig. 6.7 a LUT size equal to signal points are used (4000 points). Nevertheless, it is a high value. Usually, FPGA works with 512 or 1024 points. Working with less points, and comparing the LUT values when one iteration or 30 iterations are used (changing the amplitude of the signal sent), is represented on Fig. 6.9.

Notice how the compression gain effect is less here. The reason is that it is applied less carrier power level to the signal.

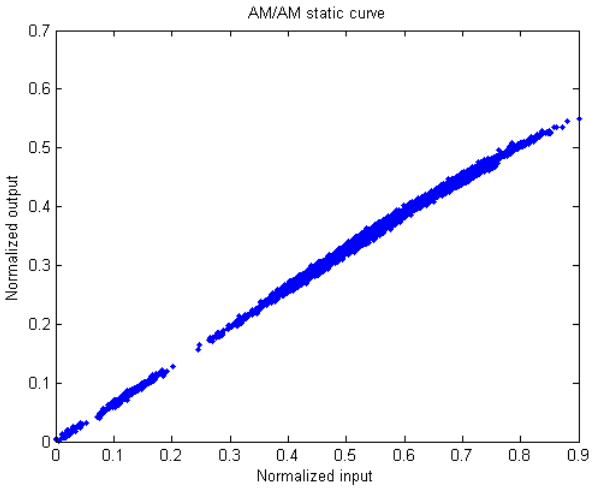


Fig. 6.8 AM/AM curve with just one iteration

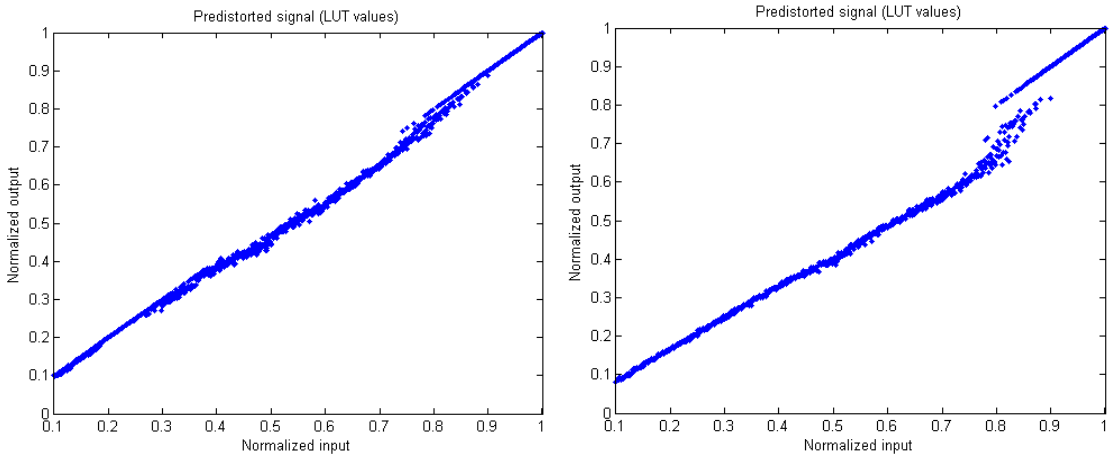


Fig. 6.9 (a) Estimated predistorter function, LUT values for one iteration (b) Estimated predistorter function, LUT values for 30 iterations

It is clearly appreciated how LUT change the values of the points that are excited by the PA signal. After 30 iterations (changing the signal amplitude), there are more points that changes its value than when just one iteration is applied.

6.3.3. Gain curve. LUT size implication.

By means of Fig. 6.10, where the PA gain curve is represented, the LUT size implication is studied.

Having a LUT size longer, obviously, more points are represented. However, if the LUT size is shorter, the effect that produces in the curve is like a mean would be done.

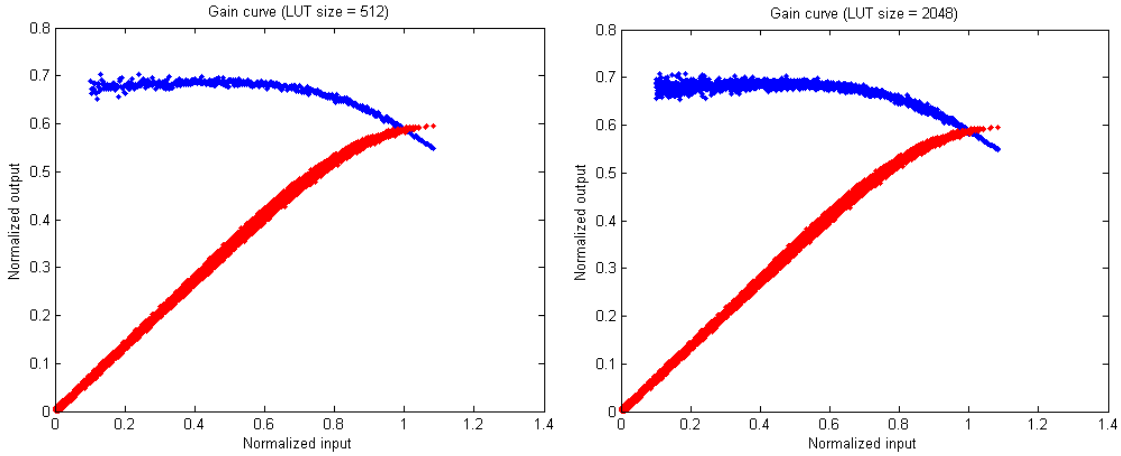


Fig. 6.10 (a) Gain function, LUT size = 512 (b) Gain function, LUT size = 2048

6.4. Adaptive predistortion results (LMS algorithm)

Now, an adaptive predistortion is implemented. The LMS algorithm is used with a μ parameter of 0.01.

6.4.1. PA identification

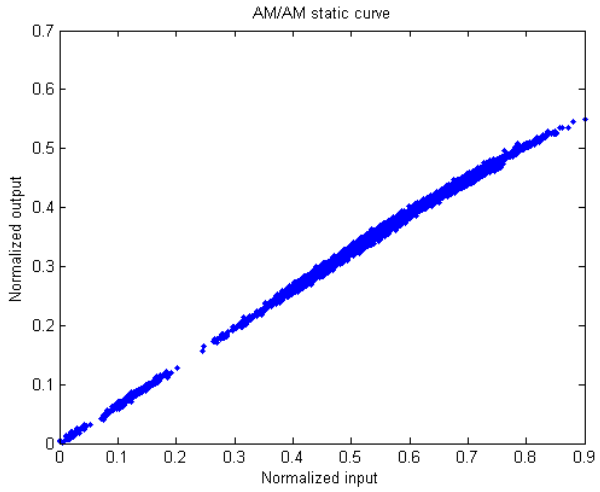


Fig. 6.11 AM/AM curve with just one iteration

6.4.2. Predistortion curve identification

Using the LMS algorithm, the LUT result is this one. It is clearly appreciated how with one iteration, the LUT is nearly all ones. It is due to the μ parameter, which is too small (0.01). So, the convergence of the LUT values is slower than the others algorithm. But using an adaptive predistortion implies a better approximation to the real function. The reason, as it has been commented before, is that this algorithm have into account the past values.

If the Fig. 6.12 (b) is observed, it can be seen how the LUT values are taking the correct predistortion function form with 100 iterations. Once the LUT have the correct values after several iterations, the convergence is better than the other algorithms used in this study.

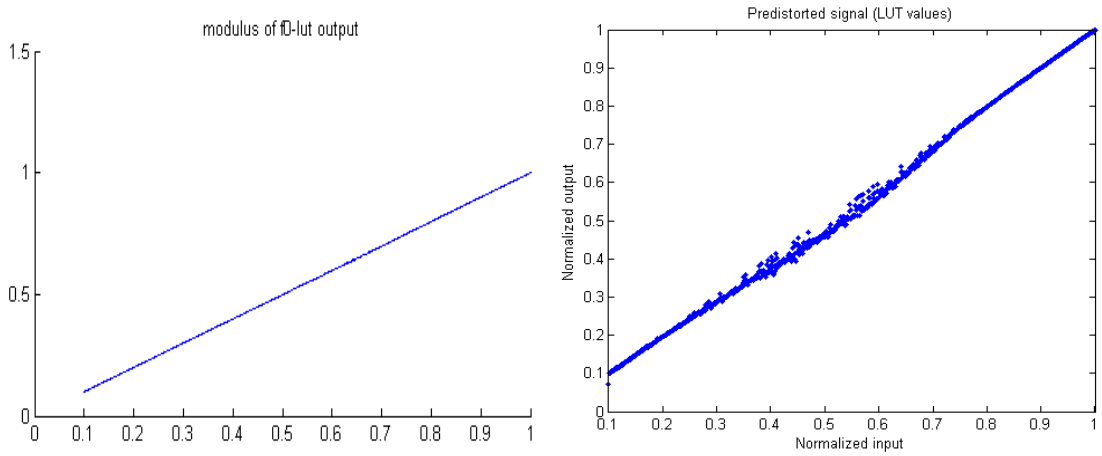


Fig. 6.12 (a) Estimated predistorter function, LUT values, one iteration. μ=0.01
(b) Estimated predistorter function, LUT values, 100 iterations. μ=0.01

CHAPTER 7. CONCLUSIONS

The goal is finally achieved. The complete system is absolutely controlled by Matlab. The signal is created from Matlab, sent to the PA and finally obtained again from Matlab, making the GPIB commands and the VSA software transparent to the user.

Regarding the VSA software, it has been proved that it is a very useful utility when a signal wants to be studied. All the information data of the signal could be shared with Matlab. Thus, it makes possible to study the overall system. Until now, it was only achievable to see the received signal in a Spectrum Analyzer and, as much, to see parameters as the constellation points, the EVM or the ACPR. Now, with the VSA, the complete information data of all points of the signal could be saved and manipulated with Microsoft Excel, ADS or Matlab.

Maybe at begin of using this software the user could be a little bit lost. But once all the VSA parameters and aspects are controlled it is easy to use. Moreover, the GPIB command, combined with this software, allows a complete control of a communication system. It is very practical for any kind of study. The platform and scenario achieved here can be used for a lot of works. For example, for prove PAs, for compare different linearization types or for any study in which ths signal wants to be considered.

On the other hand, no more PA software models are needed in order to simulate the predistortion algorithm. The measurements can be done with a real PA and, besides, the phase information could be also linearised. Working with the real components and devices under test is always better than if a software model is used.

About the digital predistortion, different conclusions are observed.

Firstly, it has to be mentioned that the digital predistortion is achieved. It is proved how it improves the nonlinearities due to the PA. Different ways to do this predistortion are used and compared here: non-adaptive without LUTs, non-adaptive with LUTs and adaptive with LUTs.

As it can be noticed at the figures showed on chapter 6, the predistortion without any type of LUT is quite “chaotic”. The outcome is resulting with blurring effects or with the points scattered. This is because the signal has 4000 points and an eight times bucle is done (changing the signal amplitude). Thus, 32000 points are finally presented.

When no LUT is used, all these 32000 points are predistortioned and any type of mean of the predistortion curve is done. However, it is clear observed how the digital predistortion correctly works. The consequence of using predistortion is that the result is linear but losing PA gain. It is important to say that applying a digital predistortion before the PA the system total gain can be selected. Notice

on Fig. 6.5 how the result (black curve) follows just right the ideal curve (red line).
On the other hand, a non-adaptive digital predistortion is done, but now with LUTs. The main advantage of using a LUT is the computational speed. Other advantage is that make a LUT is similar to make the mean of all the 32000 values in a less points (LUT size). It could be seen clearly comparing the graphs on chapter 6.2 with the graphs on chapter 6.3.

It has to be mentioned that if a FPGA implementation wants to be developed, the LUT size should be power of two (512, 1024…) and the mean of all points can be done.

With the LUT case, the result is more linear than if any LUT is used.

Finally, an adaptive predistortion holding in a LMS algorithm is performed. In this case the LUT not depends exclusively on the actual value and it also depends on the past values of the LUT. It can be seen how the actualization of the LUT depends on the μ value. When the μ parameter is small, the progression is slower than if the μ value is big. In this study, a μ parameter of 0.01 is used. It will improve the performance of the overall system.

7.1. Future work

Some works that could be done in a future with the platform Matlab – GPIB – VSA created are the followers:

- Once the VSA is completely adaptive and transparent to a Matlab user, any signal test can be done.
- For example, another algorithm to do the digital predistortion can be prove and compared with any else. For instance, the NARMA model [20].
- The comparison of several modulation signal types (4QAM, 16QAM…) or different standards (IEEE 802.16a, b, g…, UWB…) with a same PA.
- The comparison of several PA models.
- …

7.2. Environmental study

It is important to have into account the environmental study in all projects done nowadays. The future must be sustainable and it depends exclusively in what is done today.

This work talks about three main issues: software, hardware and predistortion. All these issues have some relation, in some way, with the environmental study.

- Software: Software is the less participant in environmental aspects of this work, although it has some particular topic that must be in mind. The time that the code is running is directly proportional to the energy consumed by the PC. So, the code should be efficiently created for having a less time running.

- Hardware: The hardware equipment is composed by a lot of electric pieces. All these parts must be places on the correct container when the device is broken or is obsolete.

- Predistortion: As it is said before on this work, when the predistortion is working correctly, one can work close to the compression point, achieving a longer battery life. So, if the battery life is extended implies that finally, the total energy used will be less.

BIBLIOGRAPHY

[1] http://www.rf-amplifiers.com/index.php?topic=intercept

[2] R.L. Brooker, “Spectral-Null Pulse Waveform for Characterizing Gain and Phase Distortion in Devices with Uncorrelated Frequency Translation or Limited CW Power Capability”

[3] K.Fazel and S. Kaiser, “Analysis of Non-Linear Distortions on MC-CDMA”

[4] B. Elbert and M. Schiff, “Simulating the performance of Communication Links with Satellite Transponders”

[5] M. K. Nezami, “Fundamentals of Power Amplifier Linearization Using Digital Pre-Distortion”, High Frequency Electronics, September 2004

[6] P.L. Gilabert, G. Montoro and A. Cesari, “A Recursive Digital Predistorter for Linearizing RF Power Amplifiers with Memory Effects”, Proceedings of Asia-Pacific Microwave Conference, 2006.

[7] W.J. Kim, S.P. Stapleton, J.H. Kim and C. Edelman, “Digital Predistortion Linearizes Wireless Power Amplifiers”, IEEE Microwave Magazine, pp. 54-61, September 2005.

[8] J. Vuolevi, “Distortion in RF Power Amplifiers”, Artech House INC, 2003

[9] H. Qian, L. Ding, G.T. Zhou and J.S. Kenney, “Predistortion Linearization Measurement Results for Power Amplifiers with Memory Effects”, School of Electrical and Computer Engineering Georgia Institute of Technology Atlanta, GA 30332-0250, USA.

[10] http://cnx.org/content/m10481/latest/

[11] A.C. Carusone and D.A. Johns, “Digital LMS Adaptation of Analog Filters Without Gradient Information”, University of Toronto.

[12] http://en.wikipedia.org/wiki/Lookup_table

[13] http://cp.literature.agilent.com/litweb/pdf/5989-4074EN.pdf

[14] http://cp.literature.agilent.com/litweb/pdf/5968-3386E.pdf

[15] http://en.wikipedia.org/wiki/VXI

[16] http://en.wikipedia.org/wiki/Amphenol

[17] http://en.wikipedia.org/wiki/Micro_ribbon

[18] http://www.scpiconsortium.org/scpiinfo2.htm

[19] http://sine.ni.com/nips/cds/view/p/lang/en/nid/201586

[20] G. Montoro, P.L. Gilabert, E. Bertran, A. Cesari, D.D. Silveira, “A New Digital Predictive Predistorter for Behavioral Power Amplifier Linearization”.

ANNEX

I. Supported modulation formats on the VSA

Available with Option AYA

APCO 25	DECT	DVB64	HIPERLAN/1 (HBR)	TETRA
Bluetooth™	DTV8	DVB128	HIPERLAN/1 (LBR)	VDL mode 3
CDMA base	DTV16	DVB256	NADC	WLAN (802.11b)
CDMA mobile	DVB16	EDGE	PDC	ZigBee (IEEE 802.15.4-2003)
CDPD	DVB32	GSM	PHP (PHS)	

General modulation formats, available with Option AYA

(With variable center frequency, symbol rate, filtering type and alpha/BT)

BPSK, 8PSK	VSB 8-, 16-	Offset QPSK
QPSK	FSK 2-, 4-, 8-, 16-level	EDGE
Pi/4 DQPSK	DQPSK	DVBQAM 16, 32, 64, 128, 256
MSK type 1, type 2	D8PSK	APSK 16/32 (12/4QAM)
QAM 16-, 32-, 64-, 128-, 256-, 512-, 1024-		

3G Wireless communications formats

cdma2000/1xEV-DV	Opt B7T
W-CDMA/HSDPA	Opt B7U
1xEV-DO	Opt B7W
TD-SCDMA	Opt B7X
All of the above	Opt B7N

Broadband Wireless Access formats

IEEE 802.16-2004 OFDM	Opt B7S
IEEE 802.16 OFDMA	Opt B7Y

Wireless Networking formats

WLAN (IEEE 802.11a,b,g); WLAN (HiperLAN/2)	Opt B7R
IEEE 802.11n MIMO (WLAN-HT)	Opt B7Z

Public Safety Radio formats

TETRA Enhanced Data Service	Opt BHA

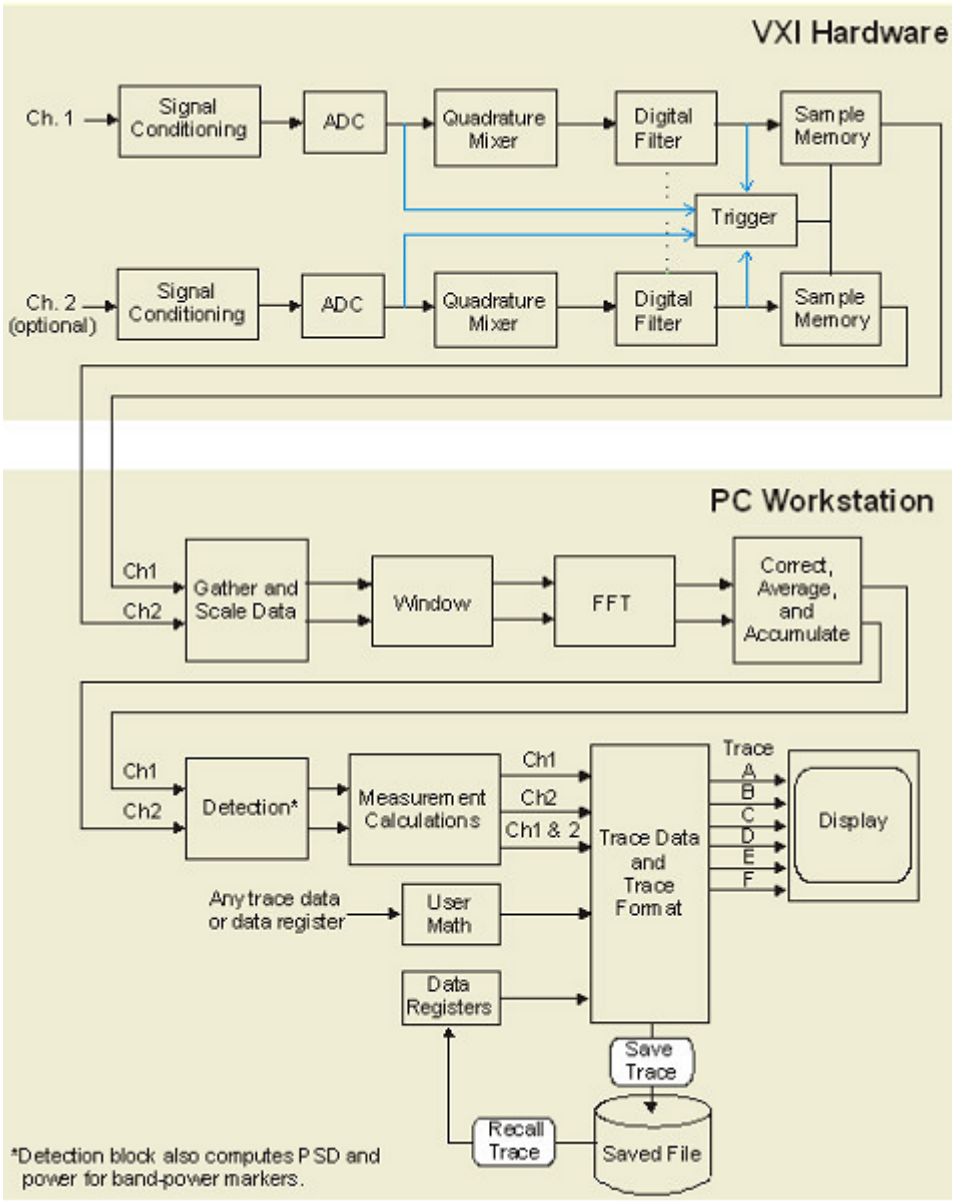
Ultra-wideband formats

MB-OFDM	Opt BHB

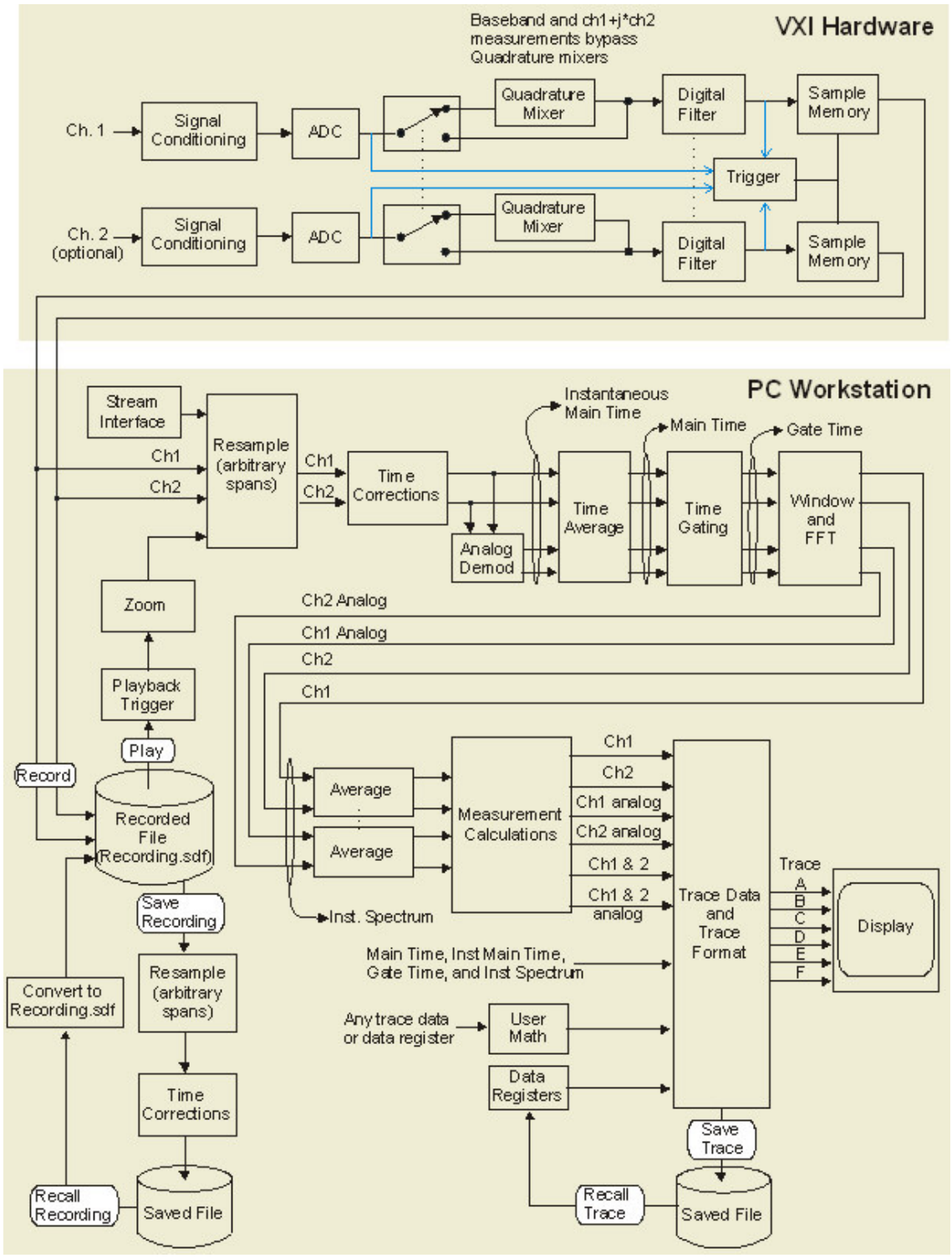
II. Supported data formats on the VSA

- Frequency response
- Correction
- Impulse response
- Inst/average Error Vector Spectrum/Time
- IQ Mag Error
- Inst/average IQ Meas Spec
- IQ Meas Time
- IQ Phase error
- Inst/average IQ Ref Spec
- IQ Ref Time
- Inst/average Spectrum
- Raw Main Time
- Search Time
- Syms/Errs
- Time

III. How the VSA makes the measurements



The following block diagram shows how the analyzer performs scalar measurements



The following block diagram shows how the analyzer performs vector measurements

IV. Pre-recorded signals on the VSA

Table 1

Signal file name	Description
128QAM.sdf	128QAM digital modulated signal; Fc @ 2 GHz, Span @ 36 MHz, 31.25 MHz SymbolRate, RRC filter with alpha = 0.22.
1xEVDOFwd.sdf	Standard 3GPP2 1xEV-DO forward link signal; Fc = 2.2GHz, Span = 1.5 MHz.
1xEVDORev.sdf	Standard 3GPP2 1xEV-DO reverse link signal; Fc = 2.2GHz, Span = 1.5 MHz.
1xEVDV.sdf	Standard 3GPP2 1xEV-DV forward link signal; Fc = 1.85625 GHz, Span = 1.5 MHz. This signal can be analyzed using cdma2000 demod mode with "Enable 1xEV-DV" selected. You can view the 16QAM modulation format used on some of the W32 code channels in the Channel traces data results.
2Ch-RF.sdf	RF signal captured through two parallel paths, Fc @ 5.805GHz, Span @ 36MHz, -5dBm and -45dBm. Use cross channel measurements and averaging to see RF channel characteristics.
2Ch-RF-Noisy.sdf	Same as the 2Ch-RF.sdf recording above, with lower SNR. Use cross channel measurements and averaging to see RF channel characteristics.
3GPPDown.sdf	Standard 3GPP W-CDMA downlink signal; Fc = 1GHz, Span = 5 MHz.
3GPPTM5H8D30.sdf	Standard 3GPP2 HSDPA downlink signal, Test Model 5, with 8 HS-PDSCH and 30 DPCH , Fc = 1.85625GHz, Span = 5MHz. This signal be analyzed in W-CDMA demod mode with "Enable HSDPA" selected. You can view the 16QAM modulation format used on some of the S16 code channels in the Channel traces data results.
3GPPUp.sdf	Standard 3GPP2 W-CDMA uplink signal; Fc = 1GHz, Span = 5 MHz
50PCAM.DAT	AM signal; 50% amplitude modulated by a 25 kHz sine wave with a 5 MHz RF carrier.
80211a_64QAM.sdf	IEEE std 802.11a/g OFDM signal with 64 QAM format; Fc = 5.805 MHz, Span 31.25 MHz.
80211b-Barker1.sdf	IEEE std 802.11b signal with Barker1 burst type and DBPSK modulation for 1 Mbps; Fc = 2.412 MHz, Span 34.375 MHz.
80211b-CCK11-short.sdf	IEEE std 802.11b signal with CCK and short PLCP header burst type and QPSK modulation for 11 Mbps; Fc = 2.412 MHz, Span 34.375 MHz.
80211g-PBCC22-short.sdf	IEEE std 802.11b signal with PBCC22 and short PLCP header burst type and 8PSK modulation for 22 Mbps; Fc = 2.412 MHz, Span 34.375 MHz.
80211n-MCS15-20MHz-MM.sdf	20 MHz Mixed Mode 802.11n signal, using MCS 15 (which means 2 data streams, each using 64-QAM data subcarrier modulation format).
80211n-MCS15-40MHz-GF.sdf	40 MHz Green Field 802.11n signal, using MCS 15 (which means 2 data streams, each using 64-QAM data subcarrier modulation format).
AMPMSQR.DAT	Carrier signal Amplitude Modulated by a square wave
Ampmtri.dat	Triangle wave, Fc = 5MHz, Span = 156.25kHz
APSK_32_9_10.sdf	32 APSK example recording with 9/10 coding rate (no Header or Pilot slots). This signal works with "Digital Video > DVB 32APSK > Code Rate 9/10" standard preset in AYA Digital Demod Analysis.
BSTIMING.DAT	RF Burst signal; Fc = 5 MHz, QPSK signal, 50 kHz SymbolRate, RRC filter with alpha = 0.35.
BSTQPSK.DAT	RF Burst, QPSK modulated signal; Fc = 5 MHz, Pi/4 DQPSK signal, 50 kHz SymbolRate, RRC filter with alpha = 0.35.
cdma2000Fwd.sdf	Standard 3GPP2 cdma2000 Forward link signal; Fc = 1 GHz, Span 2.6 MHz.
cdma2000Rev-LongCodeMask0.sdf	Standard 3GPP2 cdma2000 Reverse link signal; Fc = 1 GHz, Span 1.5 MHz. The Long Code Mask parameter needs to be set to 0.
Edge_5Mhz.dat	Standard EDGE(3p/8 8PSK) "Enhanced Data for Global Evolution" digital modulated signal; Fc = 5 MHz, Span 625 kHz.

GATE2BUR.DAT	This signal has two TDMA (time division multiple access) bursts. Both bursts are QPSK modulated at 50 kilo-symbols per second. The first burst is modulated with a random bit stream with an equalization sequence in the middle. The second burst is 10 dB lower that the first and modulated with a bit pattern of 8 ones and 8 zeros.
HiperLAN2_16QAM.sdf	Standard 3GPP2 HIPERLAN/2 OFDM signal with 16 QAM format; Fc = 1 GHz, Span 31.25 MHz.
i80216e_DL10MHz.sdf, i80216e_10MHz.set	IEEE 802.16e downlink subframe signal using a 10MHz profile. Contains PUSC zone (12 symbols) followed by a FUSC zone (10 symbols). The i80216e_10MHz.set file may be used directly to setup the analyzer for analysis of this recording. This signal was generated using the i802.16e_10MHz.xml file with Signal Studio OFDMA (N7615A version 1.2.1.0), see the i802.16e_10MHz.xml table entry for more information.
i802.16e_10MHz.xml	This is the Signal Studio setup file for the i80216e_DL10MHz.sdf (downlink) and i80216e_UL10MHz.sdf (uplink) recorded signals created using Signal Studio OFDMA (N7615A version 1.2.1.0). For signal characteristics see the table entry for each respective signal. Setup Instructions: 1) Downlink setup using the i80216e_DL10MHz.sdf: For the Downlink signal, the signal studio and analyzer setup is already fully configured, and may be generated/loaded directly from the N7615A UI. 2) Uplink setup using the i80216e_UL10MHz.sdf: For the Uplink signal, the signal studio N7615A UI setup will need the "Output Mode" parameter changed to "Uplink Only(TDD)" before being generated/loaded. The analyzer setup needs the Format Subframe type set to Uplink and the Time Manual Sync Search selected and the Sync Search Offset parameter specified to 15 symbols.
i80216e_DL10MHz_StcA_01_Impaired.sdf	The i80216e_DL10MHz_StcA_01_Impaired.sdf is an 802.16 OFDMA signal that demonstrates STC zone impairments due to antenna feedthrough. IEEE802.16e downlink subframe consisting of 3 zones. Zone 2 has STC enabled with Matrix A (This recording is Antenna 0, with feedthrough from Antenna 1, and some added noise). The demodulation auto configuration capability should correctly configure measurements for any of the three zones. In demodulation mode the "STC Info" trace shows information about the Antenna0 and Antenna1 pilots present in the signal (power, relative power, and pilot RCE). Because of the feedthrough from Antenna 1, regular RCE and data burst metrics will include the feedthrough. In a real receiver, the STC nature of the signal would be exploited to provide improved signal quality. Because the data subcarriers are unreliable when STC analysis is enabled, the channel estimation algorithm of the VSA avoids using the data subcarriers. If the "Preamble, Data, & Pilots" equalizer training is selected (the default setting), the analyzer behaves as if "Preamble & Pilots Only" is selected instead.
i80216e_DLPuscQ16.sdf, i80216e_DLPuscQ16.set	This is a Downlink subframe using the 10MHz profile defined in IEEE 802.16e. A single 16QAM data burst is defined within a single 22 symbol PUSC zone. This signal was created mathematically and contains no real noise or impairments.
i80216e_DLPuscUniformQ64.sdf, i80216e_DLPuscUniformQ64.set	This is a Downlink subframe using the 10MHz profile defined in IEEE 802.16e. A single 64QAM data burst is defined to occupy an entire 22 symbol PUSC zone. This signal was recorded and contains real noise.
i80216e_DlSeq.sdf	This is an DL (downlink) 802.16e OFDMA signal used to demonstrate the DL Auto configuration capabilities of 802.16 OFDMA Modulation Analysis. This signal contains a repeating sequence of 3 frames, each with a different zone configuration. You can use the Auto (auto-configuration) feature on the Zone Definition tab to demodulate and analyze the signal. Auto-configuration decodes the FCH and DLMAP to determine the data region geometry defined in the rest of the subframe. The “Data Burst Info”, “DL-MAP Info” and “UL-MAP Info” traces show frame information relevant to the auto-configured measurement. Also, the auto-detected geometry is shown on the zone definition tab after each measurement. The zone definition may be saved to a MapFile for later analysis by pausing the measurement and pressing the “Save To MapFile” button on the zone definition tab.
i80216e_UL10MHz.sdf, i80216e_10MHz.set	IEEE 802.16e uplink subframe signal using a 10MHz profile. Contains PUSC zone (15 symbols) followed by an OPUSC zone (6 symbols).The i80216e_10MHz.set file may be used directly to setup the analyzer to analyze this recording after changing the subframe type to Uplink. This signal was generated using the i802.16e_10MHz.xml file with Signal Studio OFDMA (N7615A version 1.2.1.0), see the i802.16e_10MHz.xml table entry for more information
i80216e_ULPuscQ16.sdf, i80216e_ULPuscQ16.set	This is an Uplink subframe using the 10MHz profile defined in IEEE 802.16e. A single 16QAM data burst is defined within a single 24 symbol PUSC zone. This signal was created mathematically and contains no real noise or impairments.
i80216e_ULPuscUniformQ64.sdf,	i80216e_ULPuscUniformQ64.sdf signal is an Uplink subframe using the 10MHz

i80216e_ULPuscUniformQ64.set	profile defined in IEEE 802.16e. A single 64QAM data burst is defined to occupy the entire 24 symbol PUSC zone. This signal was created mathematically and contains no real noise or impairments. The i80216e_ULPuscUniformQ64.set setup file will configure the analyzer.
i80216e_UlSeq.sdf	This is an UL (uplink) 802.16e OFDMA signal used to demonstrate the UL Auto configuration capabilities of 802.16 OFDMA Modulation Analysis. This signal contains a different combinations of RNG, FFB, and data-burst transmissions. Because there is no MAP information in the UL subframe, UL "Auto" auto-configuration uses statistical methods to determine burst geometry and permutation parameters for the UL subframe. The “Data Burst Info” trace shows frame information relevant to the auto-configured measurement, including detected CDMA code information. Also, the auto-detected geometry is shown on the zone definition tab after each measurement.
MBOFDM_TFC6_480Mbps.sdf (recording) MBOFDM_TFC6_480Mbps.set (setup)	The MBOFDM_TFC6_480Mbps.sdf recorded signal was generated with an Agilent Arb generator and measured with an Agilent Infiniium scope at 20 Gsa/s. The signal is nonhopped and uses TFC 6. The MBOFDM_TFC6_480Mbps.set setup file demonstrates the new Spectral Mask limit lines and ACPR features.
MBOFDM_TFC1_53.3Mbs_NoHop.sdf MBOFDM_TFC1_53.3Mbs_NoHop.set	, The MBOFDM_TFC1_53.3Mbs_NoHop.sdf signal was generated with an Agilent ADS simulation. The signal is nonhopped and uses TFC 1. The MBOFDM_TFC1_53.3Mbs_NoHop.set file turns off hopping, allowing the signal to be analyzed correctly.
P80216e_DLPusc.sdf, P80216e_DLPusc.set	This signal was generated for the 6.10 release using P802.16-2004/Cor1/D2. It is now obsolete, but may be useful with pre-existing tutorial literature. This file may be removed in later versions of the 89601. Downlink subframe containing one PUSC zone, with an FCH and 3 bursts within that zone. The FCH and Burst01 are QPSK, Burst02 is 16 QAM and Burst03 is 64QAM. A setup file has been provided to configure the VSA so that each burst may be separately analyzed.
P80216e_DLPuscUniform16Q.sdf	This signal was generated for the 6.10 release using P802.16-2004/Cor1/D2. It is now obsolete, but may be useful with pre-existing tutorial literature. This file may be removed in later versions of the 89601. Downlink subframe containing a single PUSC zone and a single 16QAM burst within that zone. The subframe is 20 symbols long and contains no FCH. The burst covers 100% of the slots within the zone, meaning that all OFDM subcarriers are occupied for the entire burst. (Select Downlink on the Format tab, click Preset to Standard and make sure Data Burst Analysis is cleared.)
P80216e_ULPuscQ16.sdf, P80216e_ULPuscQ16.set	This signal was generated for the 6.10 release using P802.16-2004/Cor1/D2. It is now obsolete, but may be useful with pre-existing tutorial literature. This file may be removed in later versions of the 89601. Uplink subframe containing a single PUSC zone and a single 16QAM burst within that zone. The subframe begins at symbol #26 of the frame and is 12 symbols long. The burst is in wrapped format and covers 1/4 of the available slots (35 slots). A setup file has been provided to configure the VSA for analyzing this signal.
P80216e_ULPuscUniformQ64.sdf, P80216e_ULPuscUniformQ64.sdf	This signal was generated for the 6.10 release using P802.16-2004/Cor1/D2. It is now obsolete, but may be useful with pre-existing tutorial literature. This file may be removed in later versions of the 89601. Uplink subframe containing a single PUSC zone and a single 64QAM burst within that zone. The subframe begins at symbol #26 of the frame and is 12 symbols long. The burst covers 100% of the slots within the zone, meaning that all OFDM subcarriers are occupied for the entire burst. When downloaded to supported Agilent signal generator (ESG, PSG, or MXG), this signal is useful for stimulus response testing of RF components. A setup file has been provided to configure the VSA for analyzing this signal. The setup file leaves the 89601A in single sweep mode so it only will take one record. For free run mode, click Control > Sweep > Continuous and then restart the measurement.
Qpsk.dat	QPSK modulated signal at 50 ksymbols/sec, root raised cosine filtered with an alpha of 0.35.
QPSKALFA.DAT	QPSK modulated signal at 50 ksymbols/sec, root raised cosine filtered with a filter alpha 0.2 instead of 0.35.
QPSKCOMP.DAT	QPSK modulated signal at 50 ksymbols/sec, root raised cosine filtered with an alpha of 0.35 with compression errors.
QPSKIBAL.DAT	QPSK modulated signal at 50 ksymbols/sec, root raised cosine filtered with an alpha of 0.35 with a 1 dB gain difference between the I and Q channels.
QPSKIOFF.DAT	QPSK modulated signal at 50 ksymbols/sec, root raised cosine filtered with an alpha of 0.35 with -22 dB Offset error.
QPSKNQST.DAT	QPSK modulated signal at 50 ksymbols/sec, Nyquist (or raised cosine) instead of a

	root-raised cosine filtering with an alpha of 0.35.
QPSKQUAD.DAT	QPSK modulated signal at 50 ksymbols/sec, root raised cosine filtered with an alpha of 0.35 and a 5 degree quadrature error.
QPSKSMRT.DAT	QPSK modulated signal at 50 ksymbols/sec, root raised cosine filtered with an alpha of 0.35 and a 1 % symbol rate error.
QPSKSPUR.DAT	QPSK modulated signal at 50 ksymbols/sec, root raised cosine filtered with an alpha of 0.35 with a spurious signal added 36 dB below the carrier and about 11 kHz below the center frequency.
SINEWPN.DAT	This is a 5 MHz sine wave with phase noise used in the "Phase Noise Measurement Example" tutorial.
TD-SCDMA_TS0-1.sdf	TD-SCDMA waveform with both uplink and downlink pilots, as well as active channels in timeslots 0 and 1.
TD-SCDMA_TS0-6.sdf	TD-SCDMA waveform with all timeslots active. A number of active channels and spreading factors are present in the different traffic timeslots.
TD-SCDMA_TS123_Mid23_NoPilots.sdf, TD-SCDMA_TS123_Mid23_NoPilots.set	TD-SCDMA signal with no pilots that uses timeslots 1, 2, and 3 and Basic Midamble 23. The TD-SCDMA no-pilot signal is demodulated by synchronizing to the midamble in the timeslots.
TEDS_CB_100k_64Q.sdf	TEDS signal for a Control Uplink slot format, 100 kHz channel bandwidth, and 64QAM modulation type.
TEDS_NBD_150k_64Q.sdf	TEDS signal for a Normal Downlink slot format, 150 kHz channel bandwidth, and 64QAM modulation type.
TEDS_NUB_25k_16Q.sdf	TEDS signal for a Normal Uplink slot format, 25 kHz channel bandwidth, and 16QAM modulation type.
TEDS_NUB_50k_16Q.sdf	TEDS signal for a Normal Uplink slot format, 50 kHz channel bandwidth, and 16QAM modulation type.
TEDS_RAB_25k_4Q.sdf	TEDS signal for a Random Access Uplink slot format, 25 kHz channel bandwidth, and 4QAM modulation type.
WiMAX_5MHz_Impaired.sdf	802.16-2004 Downlink subframes, with a nominal bandwidth of 5 MHz and guard interval 1/4. Each subframe has a Long Preamble, followed by a BPSK FCH symbol, then three data bursts. The first burst is 22 symbols of QPSK, the second is 11 symbols of 16QAM, and the last is 8 symbols of 64QAM. The FCH symbol contains only (encoded) zeros, and the 64QAM burst has incorrect amplitude for the data subcarriers. In addition, there's a little amplitude drift during the entire subframe.
WiMAX_7MHz.sdf	802.16-2004 Downlink and Uplink subframes, with a nominal bandwidth of 7 MHz and guard interval 1/4. The downlink subframe has a Long Preamble, followed by a BPSK FCH symbol, and three data bursts. The first DL burst is 10 symbols of QPSK, the second is 20 symbols of 16QAM, and the third is 50 symbols of 64QAM. The FCH symbol correctly describes these bursts. There are two separate uplink bursts in the uplink subframe, each with a Short Preamble. The first UL burst has 15 symbols of QPSK. The second has 15 symbols of 64QAM.
XMITTER.DAT	This is a recording of a FM transmitter turning on. The recording was allowed to play and then paused when the carrier appeared. This signal is used in the FM Modulated Signal Example tutorial.
ZigBee-2450.sdf	This is a burst ZigBee signal in the 2450 MHz band.

Table 2

Signal File Name	Description
dect.sdf	This is a pulsed, standard DECT format signal. To measure this signal, select the digital demod DECT preset format and set Pulse Search to OFF
gsm.sdf	This is a pulsed, standard GSM format signal. To measure this signal, select the digital demod GSM preset format.
nadc.sdf	This is a pulsed, standard NADC format signal. To measure this signal, select the digital demod NADC preset format.
pdc.sdf	This is a pulsed, standard PDC format signal. To measure this signal, select the digital demod PDC preset format.
phs.sdf	This is a non-pulsed, standard PHS(PHP) format signal. To measure this signal, turn off Pulse Search after selecting the digital demod PHS(PHP) preset format.

V. Program code

```
Creating GPIB objects

% ANALIZADOR
g18=gpib('ni',0,18);
g18.InputBufferSize=50000;
fopen(g18)
fprintf(g18, '*IDN?');
idn = fscanf(g18)

% GENERADOR
g19=gpib('ni',0,19);
g19.InputBufferSize=50000;
fopen(g19)
fprintf(g19, '*IDN?');
idn = fscanf(g19)

Creating VSA object

hVSA = actxserver('AgtVsaVector.Application');

Configuring VSG

carrier_level=-10;
cadena=[':POW:AMPL ',num2str(carrier_level),' dBm'];
fprintf(g19,cadena);
pause(2)

carrier_freq=2.010;
cadena=[':FREQ:FIX ',num2str(carrier_freq),' GHZ'];
fprintf(g19,cadena)
pause(2)

x_gpib=100*(1+i)*ones(100,1);
fprintf(g19,':SOUR:RAD:ARB:STAT OFF')
pause(2)
esg_darb(x_gpib, 'IQSIGNAL');
pause(2)
fprintf(g19,':SOUR:RAD:ARB:WAV "ARBI:IQSIGNAL"')
pause(5)
fprintf(g19,':SOUR:RAD:ARB:STAT ON')
pause(5)

fprintf(g19,':POW:ALC:STAT OFF');

x_vsg=(1+i)*ones(200,1);
loadVSG

Configuring Spectrum Analyzer

midelay=1e8;

freq_cent=2.010;
cadena=[':FREQ:CENT ',num2str(freq_cent),' GHZ'];
fprintf(g18,cadena);

for buffer1 = 1:midelay,
```

```
buffer2=2+2;
end

freq_span=40;
cadena=[':FREQ:SPAN ',num2str(freq_span),' MHZ'];
fprintf(g18,cadena)

for buffer1 = 1:midelay,
buffer2=2+2;
end

Configuring VSA parameters

hMeasurement = get(hVSA,'Measurement');

hFrequency = get(hMeasurement,'Frequency');
set(hFrequency,'Center',2.01e9);
set(hFrequency,'Span',10e6);

nsamp=4;
ResultL=100;
set(hMeasurement,'DemodConfig',2);
hDemod = get(hMeasurement,'DigDemod');
set(hDemod,'FilterAlpha',0.22);          %Alpha cosine
set(hDemod,'Format',4);                  %QPSK
set(hDemod,'MeasFilter',2);              %Root Raised Cosine
set(hDemod,'RefFilter',1);               %Raised Cosine
set(hDemod,'ResultLen',ResultL);         %Result length
set(hDemod,'PointsPerSymbol',nsamp);     %Points per symbol
set(hDemod,'SyncSearch',1);              %SyncSearch
set(hDemod,'SyncPattern','00011011000110110001101100011011000011011');
%SyncPattern or pilot message
clock=6.144e6; %VSG Clock
SymRate=clock/nsamp;
set(hDemod,'SymbolRate',SymRate);        %Symbol Rate

hDisplay = get(hVSA,'Display');
hTraces = get(hDisplay,'Traces');
hTrace1=get(hTraces,'Item',1);
hTrace2=get(hTraces,'Item',2);
hTrace3=get(hTraces,'Item',3);
hTrace4=get(hTraces,'Item',4);
hTrace5=get(hTraces,'Item',5);
hTrace6=get(hTraces,'Item',6);
set(hTrace1,'Format','vsaTrcFmtVectorIQ');
set(hTrace1,'DataName','IQ Meas Time1');
set(hTrace1,'Active',1);
set(hMeasurement,'Continuous',1);
invoke(hMeasurement,'Start');

Create waves

clc

len_sym=1000;
nsamp=4;
len=len_sym*nsamp;
clock=6.144e6;      %VSG Clock
SymRate=clock/nsamp;
M=4;
rolloff = 0.35; % Rolloff factor of filter
```

```
sincro=[0; 1; 2; 3; 0; 1; 2; 3; 0; 1; 2; 3; 0; 1; 2; 3; 0; 1; 2; 3];
%Pilot message in order to synchronize the VSA
signal=randint(len_sym-length(sincro),1,M);
signal=[sincro; signal]; %Signal to modulate
signal=[signal; signal];

constellation=[1+j*1 -1+j*1 1-j*1 -1-j*1];
modsignal=genqammod(signal,constellation);
filtorder = 80; % Filter order
delay = filtorder/(nsamp*2); % Group delay (# of input samples)
rrcfilter = rcosine(1,nsamp,'fir/sqrt',rolloff,delay);
wave_4qam=rcosflt(modsignal,1,nsamp,'filter',rrcfilter);
wave_4qam=wave_4qam(1+40:1:len+40);
wave_4qam=wave_4qam/max(abs(wave_4qam));

disp('**  creadas las ondas');


Load Matlab signal to VSG
xi=real(x_vsg);
xq=imag(x_vsg);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GPIB AND DAC SIGNAL FORMAT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
AMPLITUD=8190;
CENTRO=8192;
xi_escalada=round(xi*AMPLITUD+CENTRO);
xq_escalada=round(xq*AMPLITUD+CENTRO);

clear buffer1;
clear buffer2;
buffer1=dec2hex(xi_escalada,4);
buffer2(:,1)=buffer1(:,3);
buffer2(:,2)=buffer1(:,4);
buffer2(:,3)=buffer1(:,1);
buffer2(:,4)=buffer1(:,2);
xi_gpib=hex2dec(buffer2);

clear buffer1;
clear buffer2;
buffer1=dec2hex(xq_escalada,4);
buffer2(:,1)=buffer1(:,3);
buffer2(:,2)=buffer1(:,4);
buffer2(:,3)=buffer1(:,1);
buffer2(:,4)=buffer1(:,2);
xq_gpib=hex2dec(buffer2);

x_gpib=xi_gpib+i*xq_gpib;

fprintf(g19,':SOUR:RAD:ARB:STAT OFF')

midelay1000;

esg_darb(x_gpib, 'IQSIGNAL');

midelay1000;
midelay1000;
midelay1000;
```

```
fprintf(g19,':SOUR:RAD:ARB:STAT ON')

midelay1000;
midelay1000;
midelay1000;
Initialize LUTs

f0_in=[0.1:0.001:1]';
f0_gain=ones(length(f0_in),1);
f0_contador=zeros(length(f0_in),1);

f1_in=[0.1:0.001:1]';
f1_gain=ones(length(f1_in),1);
f1_tau=0;

g1_in=[0.1:0.001:1]';
g1_gain=ones(length(g1_in),1);
g1_tau=0;

disp('**  inicializadas las LUTs');

Predistortion

x_dpd=x_gen*amplitude_wave;
disp('**  INICIO predistortion');

switch PD_type

case 1
disp('**  haciendo la DPD tipo 1 (promedio de luts)');
for n=1:length(x_dpd)
[valor_f0,indice_f0]=min(abs((f0_in-abs(x_dpd(n)))));
y_dpd(n,1)=f0_gain(indice_f0)*x_dpd(n);
end
%%%%%%%%%%%%%%%%%%%%%%%%%
case 2
disp('**  haciendo la DPD tipo 2 (lut-LMS)');
for n=1:length(x_dpd)
[valor_f0,indice_f0]=min(abs((f0_in-abs(x_dpd(n)))));
y_dpd(n,1)=f0_gain(indice_f0)*x_dpd(n);
end
%%%%%%%%%%%%%%%%%%%%%%%%%

otherwise
disp('Unknown method.')
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp('**  FINAL de la predistortion');

Update LUTs

disp('**  INICIO update');

AM_in=abs(x_ampli);
PM_in=angle(x_ampli);
AM_out=abs(y_ampli);
PM_out=angle(y_ampli);
deltaPM_out=PM_out-PM_in;
%
```

```
% amplificacion deseada
ampli_desired_gain=0.6;
%
switch PD_type

case 1
disp('**  haciendo la DPD tipo 1 (promedio de luts)');
for n=1:length(x_dpd)
[valor1,indice1]=min(abs(f0_in-abs(x_dpd(n))));
[valor2,indice2]=min(abs(y_ampli-ampli_desired_gain*x_dpd(n)));
if abs(x_dpd(n))>=0.1
    f0_contador(indice1)=f0_contador(indice1)+1;

f0_gain(indice1)=0.5*f0_gain(indice1)+0.5*(x_ampli(indice2)/x_dpd(n));
end
end

case 2
disp('**  haciendo update de la DPD tipo 2 (lut-LMS)');
mu_f0=0.01;
for n=1:length(x_dpd)
[valor_f0,indice_f0]=min(abs(f0_in-abs(x_dpd(n))));
buffer_ampli_gain=y_ampli(indice_f0)/y_dpd(indice_f0);
error=ampli_desired_gain*x_dpd(indice_f0)-y_ampli(indice_f0);
f0_gain(indice_f0)=f0_gain(indice_f0)+mu_f0*error*(buffer_ampli_gain*x
_dpd(indice_f0))';
end

otherwise
disp('Unknown method.')
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
disp('**  FINAL del update');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
```

VI. GPBI cable datasheet

GPIB Controller for USB 2.0 High-Speed

NI GPIB-USB-HS

- Completely IEEE 488.2 compatible
- Controls up to 14 GPIB instruments
- Compact size and light weight
- Plug-and-play configuration
- No external power required
- Built-in 2 m USB cable
- No GPIB cable required to connect to instruments
- USB 2.0 high-speed compliant
- Maximum GPIB transfer rates
  - More than 1.8 MB/s (IEEE 488.1)
  - More than 7.2 MB/s (HS488)

Operating Systems
- Windows 2000/XP

Recommended Software
- LabVIEW
- LabWindows/CVI
- Measurement Studio

Driver Software (included)
- NI-488.2

NEW

Overview

The compact National Instruments GPIB-USB-HS transforms any computer with a USB port into a full-function, IEEE 488.2 controller that can control up to 14 programmable GPIB instruments. The small size and light weight of the NI GPIB-USB-HS make it ideal for portable applications using a laptop computer or other applications in which the computer has no available internal I/O slots. The GPIB-USB-HS works with Windows 2000/XP computers with a USB port.

The GPIB-USB-HS is easy to install and use because there are no external DIP switches and you do not need to restart your computer for the system to recognize your IEEE 488.2 interface. The GPIB-USB-HS is a plug-and-play interface that the OS automatically recognizes and configures as soon as you physically attach it to the USB port on your computer. With the GPIB-USB-HS, you can get up and running quickly, so you can focus on developing your instrument control applications.

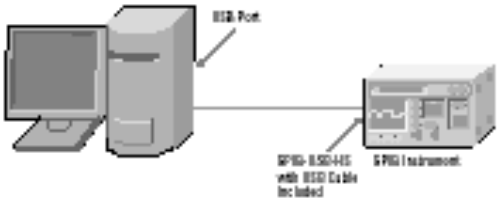


Figure 1. Easily connect your GPIB instruments to the USB port of your computer.

The GPIB-USB-HS is the first GPIB interface to take advantage of the superior performance of high-speed USB 2.0 signaling (480 Mb/s). Plugging the GPIB-USB-HS into a USB high-speed port provides industry-leading GPIB performance using both the standard and high-speed IEEE 488.1 handshake.

Using a TNT family Talker/Listener/Controller IEEE 488.2 ASIC, the GPIB-USB-HS implements the full range of GPIB controller functions, including those required and recommended by IEEE 488.2. It also implements normal and extended talker and listener, serial and parallel polling, service requests, and pass/receive control functions. Drawing power directly from the USB port, the GPIB-USB-HS requires no external power input.

With NI-488.2, you get a robust driver with additional utilities and wizards that help you troubleshoot your applications and decrease your development time (see Figure 2). Furthermore, you maintain compatibility with existing systems. Applications previously written for other National Instruments GPIB controllers can run unmodified with the GPIB-USB-HS.

Connecting the GPIB-USB-HS to Your Instruments

The GPIB-USB-HS does not require a GPIB cable for connecting to your instruments. You can attach it directly to the GPIB port on your instrument and then connect the USB cable to the USB port on your computer. If you have multiple instruments in a daisy chain or star configuration, attach any cables that connect to the other instruments first, and then piggyback the GPIB-USB-HS as the last connector in the stack.

NATIONAL INSTRUMENTS

GPIB Controller for USB 2.0 High-Speed

A. Run the Getting Started Wizard

B. Communicate with your instrument

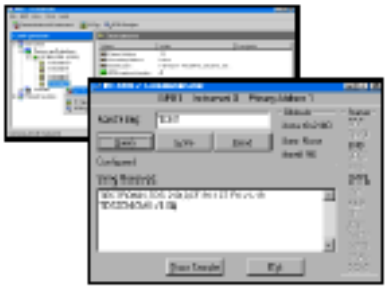


Figure 2. Take three easy steps to get up and running with your instrument connectivity.

Ordering Information

NI GPIB-USB-HS.....778927-01
Includes NI-488.2 software and a built-in 2 m USB cable.

BUY NOW!

For complete product specifications, pricing, and accessory information, call (800) 813 3693 (U.S. only) or go to ni.com/gpib.

Specifications

USB Port
High-speed USB signaling.....480 Mb/s

IEEE 488 Compatibility.....IEEE 488.1 and IEEE 488.2

Maximum IEEE 488 Data Transfer Rates[1]
IEEE 488 interlocked handshake.....1.8 MB/s
IEEE 488 overlapped
handshake (ECAT).....7.2 MB/s

[1] Actual rates depend on system configuration, instrument capabilities, and USB port rates.

External Indicators
Ready
Green.....USB full-speed
Amber.....USB high-speed
Active
Green.....Communicating

Power Requirement
USB bus-powered device
Maximum power consumption.....500 mA

Physical
Dimensions.....10.7 by 6.6 by 2.5 cm (4.2 by 2.6 by 1.0 in.)

I/O Connectors
GPIB.....IEEE 488 standard 24 pin
USB.....USB standard series B plug

Operating Environment
Temperature.....0 to 55 °C
Relative humidity.....10 to 80%, noncondensing

Storage Environment
Temperature.....-20 to 70 °C
Relative humidity.....5 to 95%, noncondensing

Compliance
Online at: ni.com/certification

VII. PA datasheet

High IP3

Low Noise Amplifier

50Ω 1400 to 2300 MHz

NEW!

ZRL-2300

CASE STYLE: FJ893
PRICE: $119.95 ea. QTY (1-9)

Maximum Ratings

Operating Temperature	-40°C to 85°C case
	-40°C to 60°C ambient
Storage Temperature	-55°C to 100°C
DC Voltage	+17V max.

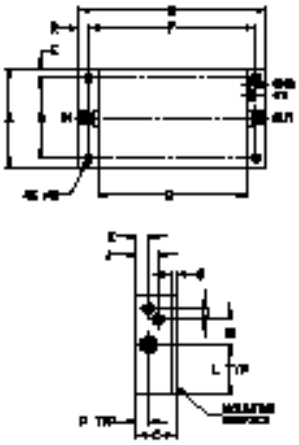
Features

- high IP3, +46 dBm typ.
- low noise figure, 2.5 dB typ.
- broad band flat gain response
- internal voltage regulated
- over voltage and transient protected

Applications

- high dynamic range
- PCS, UMTS, GSM, cellular, wireless data
- defense and satellite communications
- high linearity driver amplifier

Outline Drawing



Outline Dimensions (inch / mm)

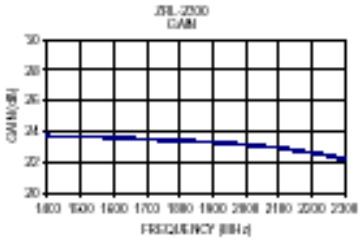
A	B	C	D	E	F	G	H	J
2.00	3.75	.80	3.80	.19	3.374	1.624	.156	.44
50.80	95.25	20.32	76.20	4.83	85.70	41.25	3.96	11.18

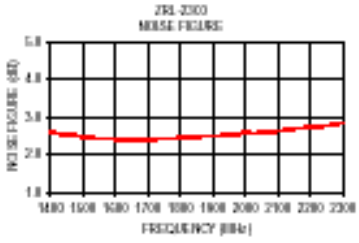
K	L	M	N	P	Q	R	S	wt.
.26	1.80	.51	.22	.26	—	.19	.10	grams
6.60	45.72	12.95	5.59	6.60	—	4.83	2.54	136.

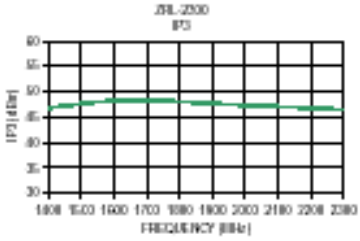
Electrical Specifications (T_{AMB}=25°C)

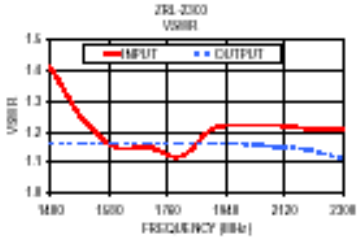
FREQ. RANGE (MHz)	GAIN (dB)				NOISE FIGURE (dB)		MAXIMUM POWER (dBm)			INTERCEPT POINT	VSWR (:1) Typ.		DIRECTIVITY (dB)	DC POWER	
			Flatness				Output (1 dB Comp.)		Input (no damage)	IP3 (dBm)					
f1 - f2	Typ.	Min.	Typ.	Max.	Typ.	Max.	Typ.	Min.		Typ.*	In	Out	Typ.	Volt (V) Typ.	Max. Current (mA)
1400-2300	23.5	21	±0.7	±1.0	2.5	3.5	24.0	23	+18	46	1.20	1.16	21	12	650
1650-2150	24	22	±0.5	±0.8	2.3	3.5	24.0	23	+18	46	1.20	1.16	20	12	650

* 1MHz tone spacing
- Other voltages available in the 6.5 to 17V range, please contact factory.

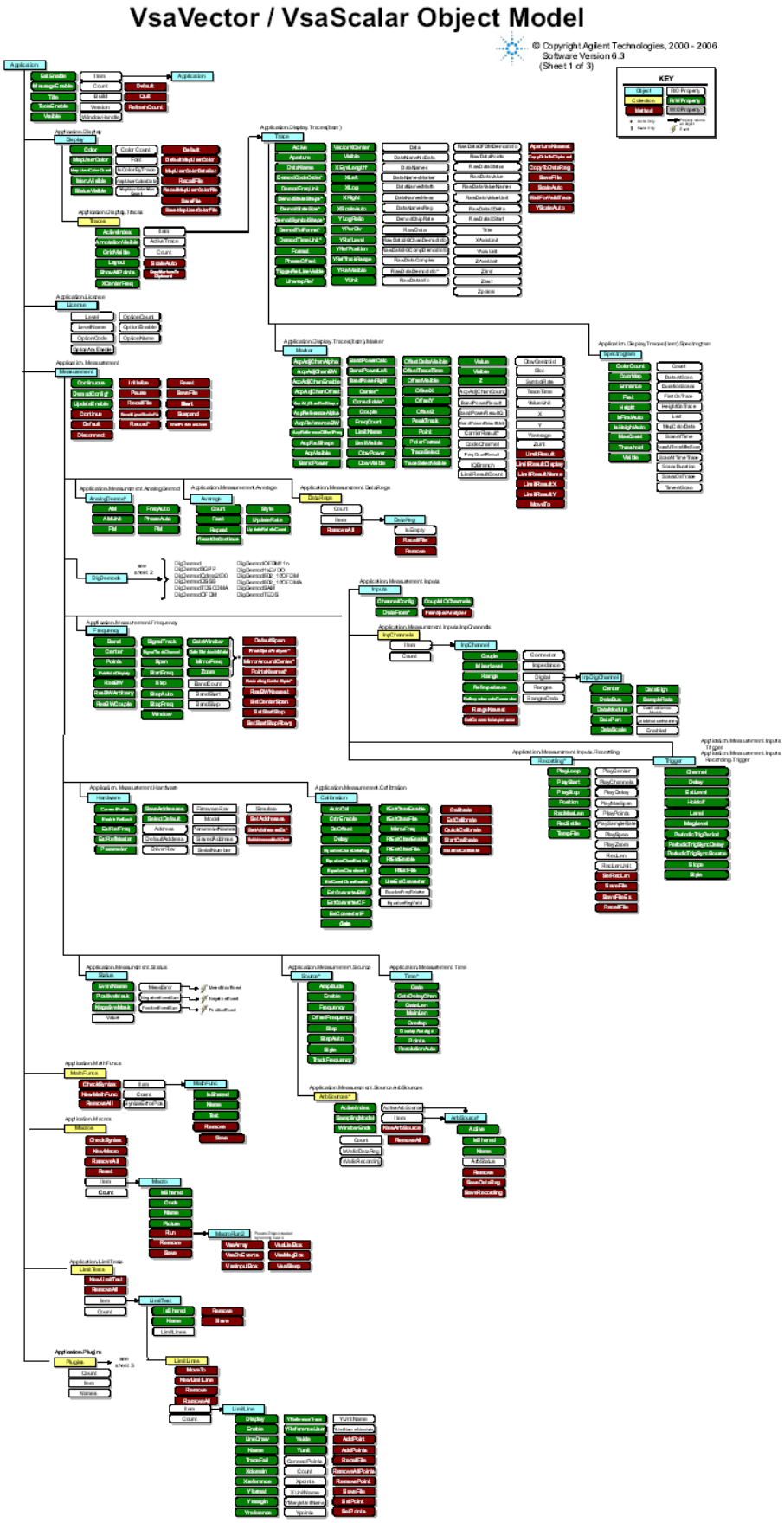






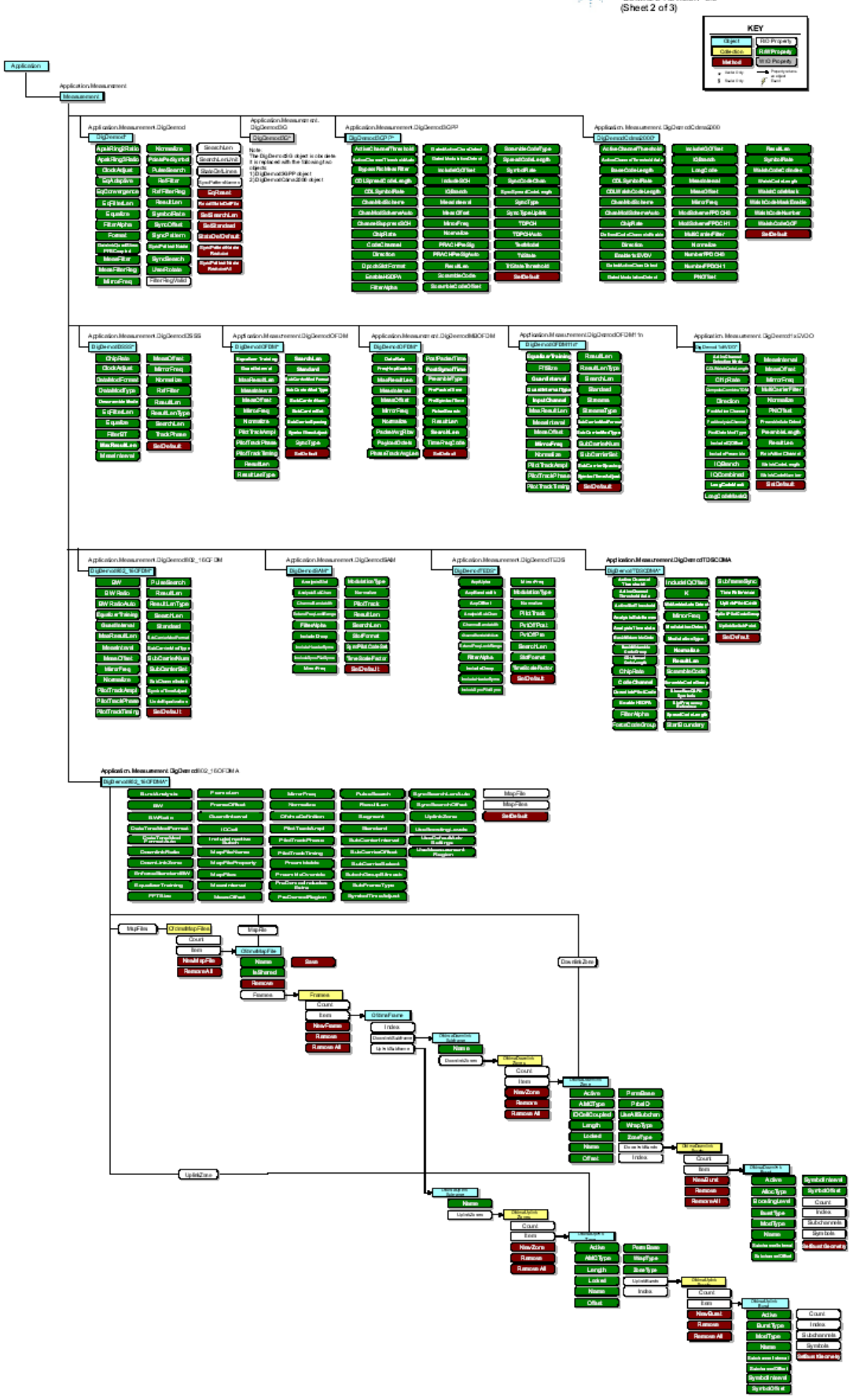


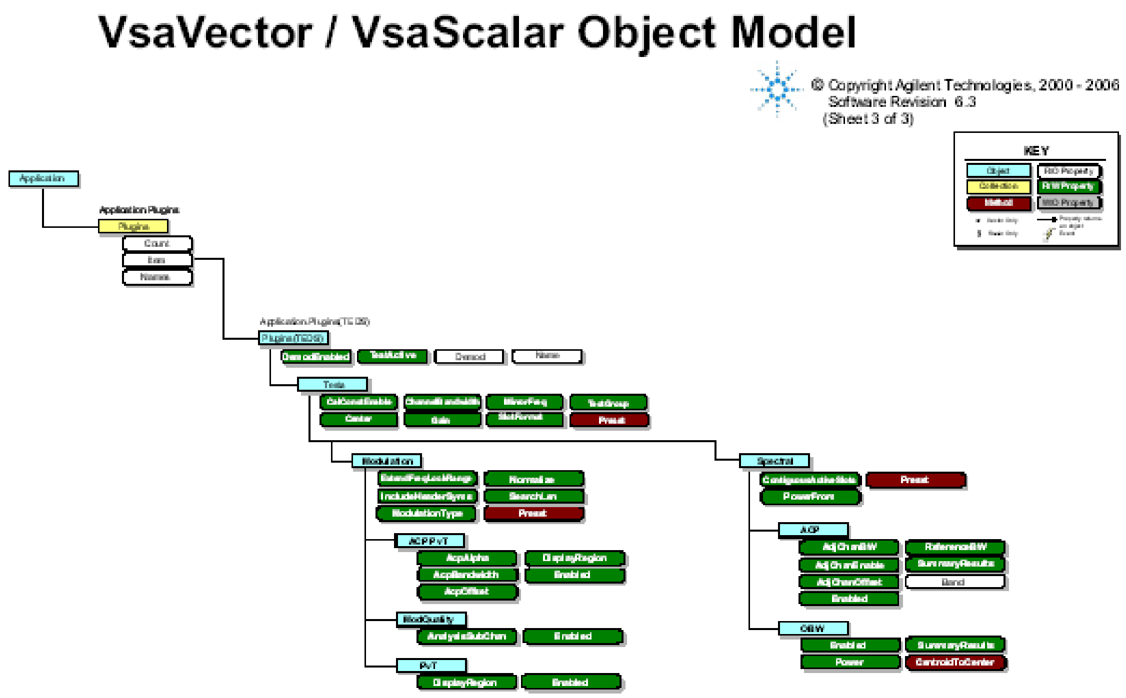
VIII. VSA object programming tree



VsaVector / VsaScalar Object Model

Software Revision 6.3
(Sheet 2 of 3)




VsaVector / VsaScalar Object Model
© Copyright Agilent Technologies, 2000 - 2006
Software Revision 6.3
(Sheet 3 of 3)
KEY
Object
Collection
Method
R/O Property
R/W Property
W/O Property
Application
Application.Plugins
Plugins
Count
Item
Names
Application.Plugins(TEDS)
Plugins(TEDS)
DemodEnabled
TestActive
Demod
Name
Tests
CalConstEnable
MirrorFreq
TestGroup
Center
Gain
SlotFormat
Preset
Modulation
ExtendFreqLockRange
Normalize
IncludeHeaderSyms
SearchLen
ModulationType
Preset
ACPPvT
AcpAlpha
DisplayRegion
AcpBandwidth
Enabled
AcpOffset
ModQuality
AnalysisSubChan
Enabled
PvT
DisplayRegion
Enabled
Spectral
ContiguousActiveSlots
Preset
PowerFront
ACP
AdjChannelBW
ReferenceBW
AdjChannelEnable
SummaryResults
AdjChanOffset
Band
Enabled
OBW
Enabled
SummaryResults
Power
CentroidToCenter