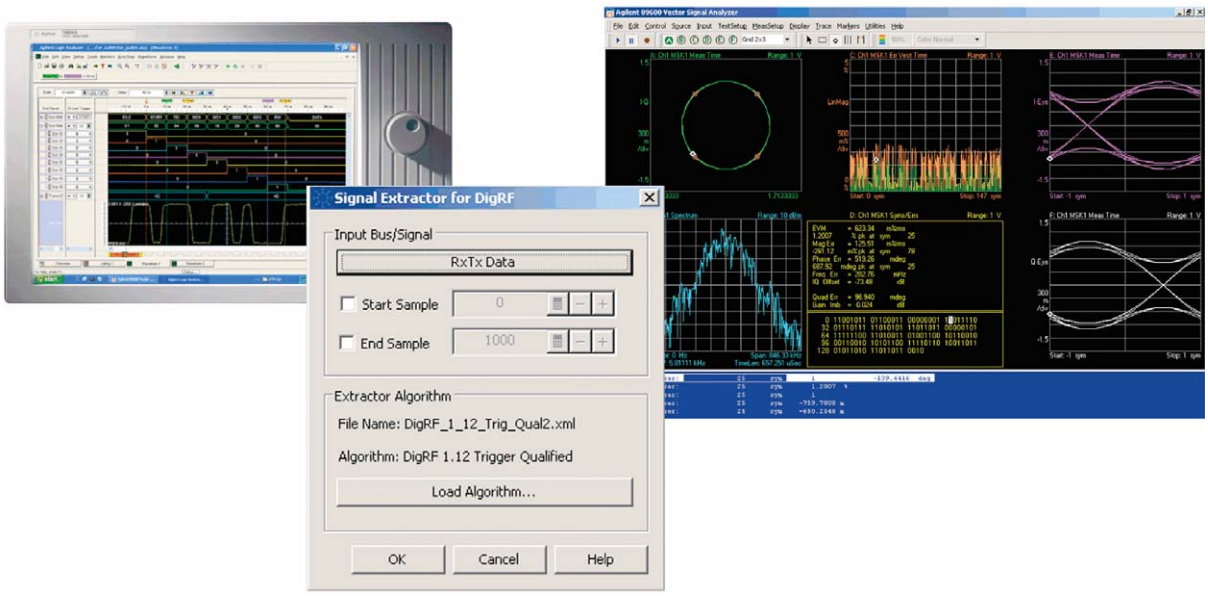


Making RF Measurements on Digital Serial Data with Agilent's Signal Extractor and the 89601A Vector Signal Analyzer

Application Note 1593

Agilent Technologies

Table of Contents

Introduction

This application note is designed to guide you through making measurements on serial data using an Agilent logic analyzer with the B4602A Signal Extractor tool and Agilent's 89601A vector signal analyzer (VSA) software. Specifically, we will look at the serial IQ interfaces from a DigRF 1.12 signal and use a preset algorithm from the B4602A tool to extract the IQ data for demodulation and logic analysis. Whether you are a digital engineer faced with making RF-type measurements or an RF engineer trying to make digital measurements, this note will help you understand how to accomplish your measurement goal.

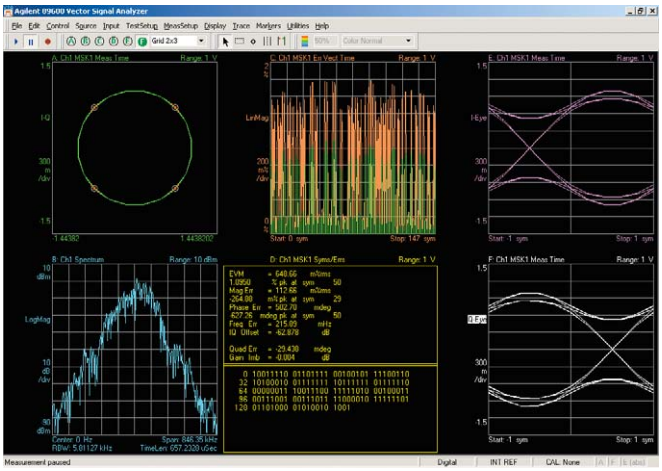


Figure 1. 89601A vector signal analysis software gives you multiple ways to view your data.

Introduction

The vector signal analyzer is a powerful tool for time, frequency and modulation domain measurements. Modulation domain measurements such as EVM, constellation and eye diagrams give you a good look at what your devices are really doing. Error vector time displays offer you insight into such things as filtering problems, distortion and frequency errors. Eye diagrams show you symbol timing that can reveal errors in clocking. The symbols/error summary table gives you a variety of information including IQ offset, phase, and magnitude error as well as EVM – it even shows you which symbols are causing the most trouble. For pulsed signals, you can look at pulse shape and timing. In all cases, you can directly correlate between the logic and RF measurements for digital IQ signals.

The logic analyzer tells you quite a bit, too. Are the levels correct? Are the right number of bits there and are they in the correct order? The logic analyzer allows you to directly correlate the data sent with the data received. You also can switch to timing mode to ensure the edges of clocks and bits are lined up correctly. Using the VSA with the logic analyzer allows you to check the accuracy of the modulator, filters and more.

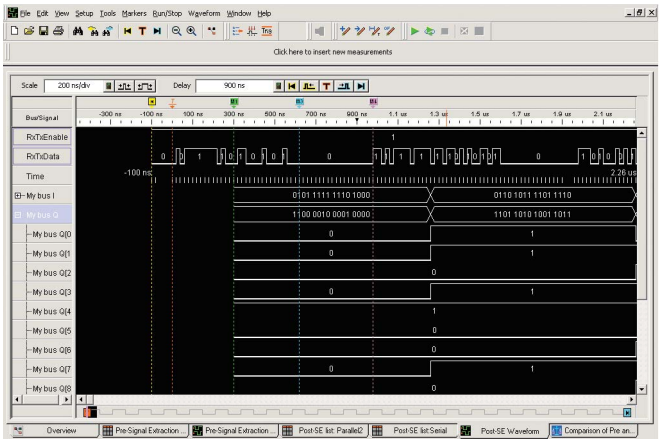


Figure 2. Agilent's logic analysis interface waveform view gives us a good picture of the relationships between the signals as well as a look at the waveforms themselves.

Introduction

Digital signals replacing analog

Digital signals are taking the place of analog in many areas. Traditionally, analog interfaces between RFICs and BBICs were standard. Now, chip manufacturers are producing chipsets with digital interfaces. Standards such as OBSAI and CPRI for base station chips and DigRF for mobile chips define a variety of digital interface possibilities. WiMedia defines a digital interface between the MAC and PHY chip. And it doesn't stop there – these types of digital interfaces are becoming common in software-defined radios (SDRs), satellite communications and radar systems, to name a few. This is a trend that is going to continue.

It makes sense – digital signals are easier and cheaper to implement and they will eventually enable better signal quality and lower power consumption. When such signals are implemented serially, they can also reduce the number of required IC pins, which saves chip makers money, and the saving typically is passed through to the end customer of consumer products.

The transition of analog to digital has not, however, changed the physical layer test requirements, and traditional analog test and measurement equipment is not suitable for digital RF design validation. Up until recently, this presented a problem. For example, how are we to make an EVM measurement off digital signals coming from an FPGA? The good news is that there is a way to do this using Agilent's logic analyzer tools with the 89601A vector signal analysis software, or the "digital VSA" as it's often called. Digital IQ signals from any logic device that can be connected to a logic analyzer can be measured in time, frequency and modulation domains.

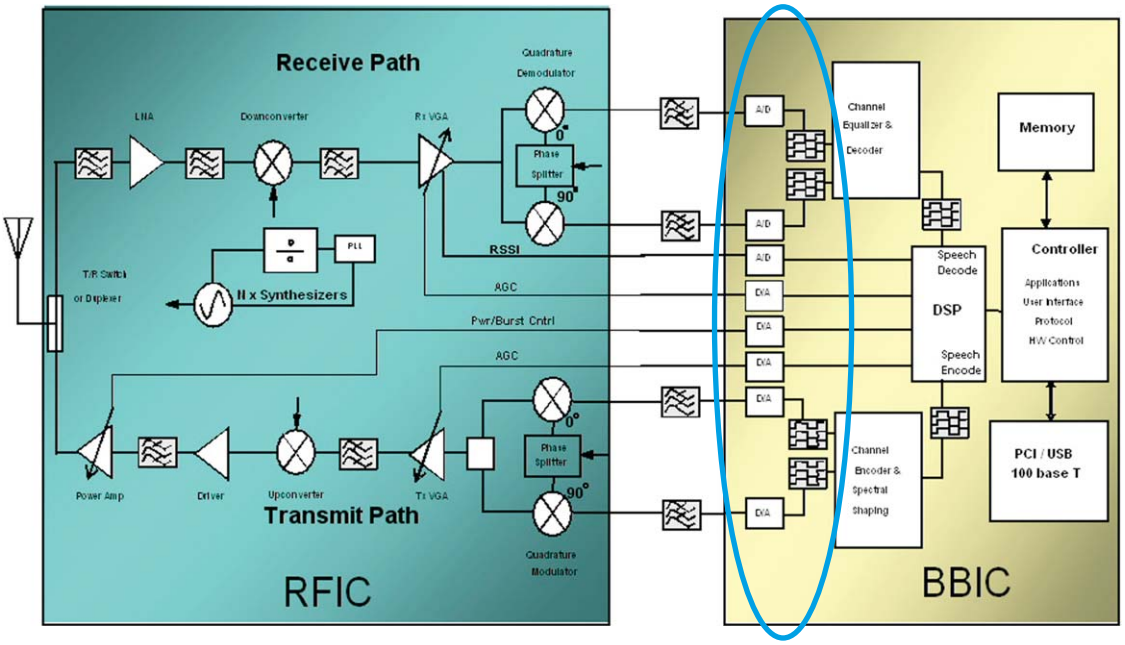
Interface changes

The transition from analog to digital interfaces does not involve a great deal of architectural change, but it does involve a move of the ADC/DAC circuitries from the RFIC to the BBIC. In some cases, the digital filtering is moved from the BBIC to the RFIC.

Serial or parallel?

Originally the digital interfaces were parallel – indeed, many still are. However, pin count on an IC is a precious resource. The average pin count for a digital parallel IQ interface is about 32 pins (not including the clocks) for a 16-bit I and 16-bit Q signal. Reduction of pin count is therefore a serious matter. Multiplexing turns out to be an easy way to implement the serial bit stream instead. This allows the signals to share common pins and greatly reduces pin count. Digital I and Q signals can be put into a serial format with some kind of time division multiplexing scheme. In some cases, as in DigRF Version 3, control data is multiplexed in, making measurements a little trickier.

The ADCs used to be located in the BBIC and the chip interface was analog.



Modern design has the ADCs placed in the RFIC, making the chip interface digital.

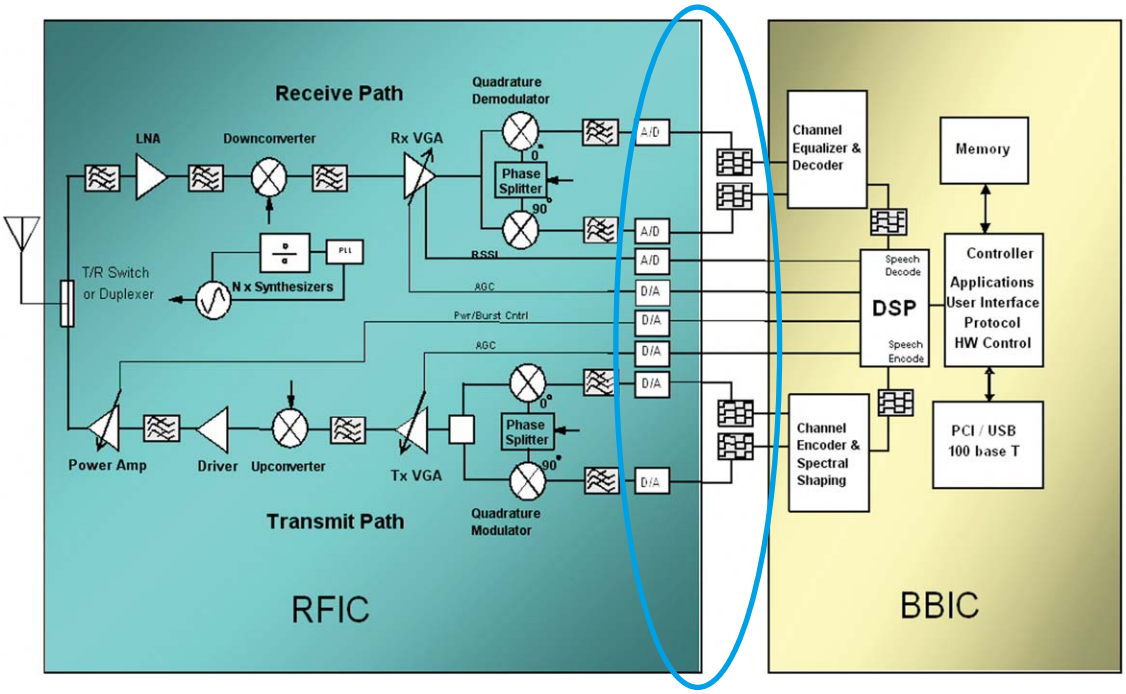


Figure 3. Changing interfaces

Measurement Basics

Logic analyzer basics

Many engineers entering the new world of digital signals are not very familiar with logic analyzers. Logic analyzers are not like spectrum analyzers or power meters or any of the standard analog test and measurement equipment. They are capable of performing very complex measurements on digital signals, and at first glance, they can look quite intimidating. The term "digital signals" refers to signals that take on a value of either 1 or 0, regardless of the type of logic (TTL, CMOS etc.) used. Logic analyzers are not as intimidating as they sometimes seem to new users at first glance. Once you become comfortable with the instrument, you'll realize that it's really quite easy to use.

The role of the logic analyzer:

A logic analyzer is a tool that gives you insight into the operation of a *digital* circuit by

- Connecting to your DUT (device under test)
- Capturing and storing the digital waveforms
- Analyzing the stored data and displaying the results

With digital signals, the logic analyzer can:

- Record a circuit's logic levels over time, and let you examine the record
- Show whether or not a particular event happens (the trigger)
- Provide a precise measure of time between events
- Inverse-assemble a microprocessor's logic levels to tell you what code was running
- Analyze complex buses and protocols

A logic analyzer is the perfect tool for connecting with the 89601A software for time, frequency and modulation domain measurements on digital IQ signals.

There are three steps to the logic analyzer process:

- Connect
- Acquire
- Display data

Connect:
You have a few choices for connecting your logic analyzer to your DUT. First, there are general-purpose probes such as the flying lead probes. These probes have individual pin connectors. They are not always the best solution because sometimes there are signal integrity issues when you connect wire to wire, but they can be very handy when there is no other way to probe. They can also be used when it is necessary to get physically close to the DUT, as is the case in DigRF.

Another way to connect involves: building a "footprint" into the board ahead of time. It's fairly common to enable probing this way: think about digital debug ahead of time, plan for it, and design connectors into the board. This is no different from putting test points onto an RF board (SMAs or SMBs) that can be easily hooked up to a cable and spectrum analyzer. The footprint is a little bigger, but there are many digital signals instead of a single RF signal. A Samtec connector is an example of a solid connector that requires a footprint to be built into the board prior to fabrication.

Figure 4. Flying leads

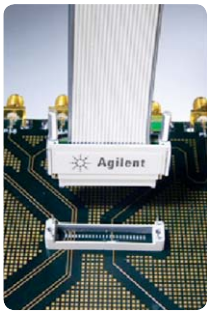


Figure 5. Samtec connector

Acquire:
There are two measurement modes in logic analysis: timing and state. Timing analysis is for logic timing; it shows you when a logic event transitions. Timing analysis is asynchronous and uses the logic analyzer clock to determine when signal transitions occur relative to one another. With state analysis, we look primarily at events themselves: it shows where you would look for events that happened, as opposed to timing relationships. In state mode, the clock comes from the DUT. While we can certainly use timing mode to check signal transitions, when you use a logic analyzer with the 89601A software, we must using state mode. We are concerned with the bits (or samples as they are known in the logic analyzer world) and the order in which they appear.

Measurement Basics

Logic analyzer basics *(continued)*

Display data:
In state mode, we are interested in seeing the data, and there are two ways of viewing it. State analysis uses the clock on the device under test as the sampling clock. This means that data is only sampled when it is valid, which just means the data is not transitioning. (We look at transitioning signals in timing mode.) The output in state mode is a listing of the values that crossed the bus, with the option of a time stamp noting when the values occurred. The state listing window will tell you what the values were going across the bus at the time the clock transitioned. This is used to track functional problems and code flow. In the waveform view, we can see the actual data on the bus.

Vector signal analyzers basics

The vector signal analyzer (VSA) is an FFT analyzer. FFT analyzers are "block" processors. They take in chunks of data from memory. This means the instrument takes in time data and performs FFTs to give the frequency domain data. It retains phase and amplitude data. The 89601A software is the FFT analyzer, and the retention of phase and amplitude data allows us to view both digital and RF signals in the RF and demod domain. When an FFT analyzer takes in data, it pulls it first from sample memory. This sample memory resides in the logic analyzer. The samples are transferred from the logic analyzer memory by the logic analyzer software, and out to the 89601A software for post-processing. Of course, when the Signal Extractor is used, only pertinent samples are transferred.

The speed with which the data is transferred to the software is dependent on the configuration you use. It is generally faster when both the logic analysis software and the VSA software are running on a common computer – it can be a separate computer, or it can be the logic analyzer's internal computer. The VSA is also a recording device, as is the logic analyzer. Whether or not the VSA is used as a recording device is up to you. However, it is necessary to understand time relationships in an FFT analyzer before it can be used as such. The 89601A software contains an extensive tutorial on these relationships, and this tutorial can be viewed without a license.

Measurement Basics

What kinds of measurements do we need?

We will use a DigRF 1.12 signal as an example. The DigRF 1.12 standard defines the digital interface between the BBIC and RFIC for GSM, GPRS and EDGE. Appendix A contains a summary of the standard for readers who are unfamiliar with it. The signal of interest to us is the RxTx data. This signal contains I and Q data, multiplexed with padding. The padding (inserted zeros) is typically used as a way to ensure proper timing. Adding padding guarantees that there are always samples available for each clock cycle. The number of padded bits used is dependent on the number of bits used for I and Q and the clock rate. Appendix A explains this in more detail. In this example, we will use a DigRF 1.12 Rx mode signal with GSM only. This means the signal is traveling from the RFIC to the BBIC, and the IQ data contains only GSM modulation.

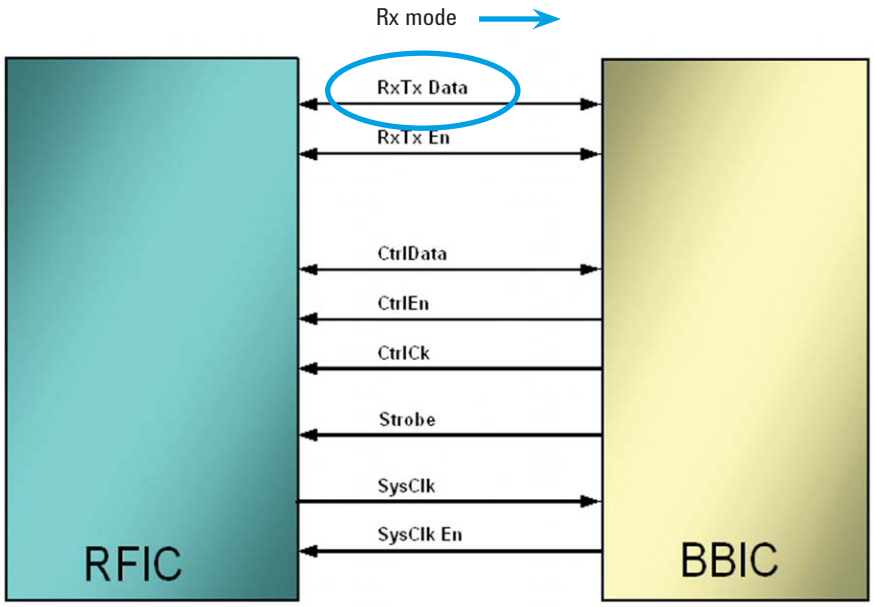


Figure 6. DigRF 1.12 interface between the RFIC and BBIC

Measurement Basics

What kinds of measurements do we need? *(continued)*

We can look at several scenarios. First, we do need a way to isolate the IQ data from the padding in the serial stream. Then we need to present the IQ data in a format that is understood by the 89601A software. Once this is done, we can look at the types of measurements required by the standard (for instance, GSM) – typically modulation quality measurements. Examples of this would be error vector magnitude (EVM), frequency error, and phase error. We may also want to make proof-of-concept measurements. In other words, we might want to look at the spectrum to ensure we have no DC offset and that the spectral shape is correct, etc.

Terminology differences:
It is important to note that the term "sample" means something different in the logic analyzer realm than it does in the DigRF specification. In logic analyzer parlance, a sample refers to a portion of a signal at a specific clock cycle. The state of a digital signal at a particular clock cycle is stored as a "sample" in memory. In the DigRF 1.12 spec, a sample refers to something a little different. Before we explain the difference, we must first understand what a symbol is.

Symbol: In IQ modulation, each "piece" of data is transmitted as a symbol. A symbol contains one "I" value and one "Q" value. Each I or Q value can represent one bit, or it can represent many bits – it's entirely dependent on the modulation format. In the DigRF 1.12 specification, there are four bits per symbol. Although GMSK only uses one bit per symbol, 8PSK uses three bits per symbol. However, the DigRF 1.12 specification uses four bits per symbol as standard, and the differences between bits per symbol for each modulation type are accounted for in the RFIC, which contains the GMSK and 8PSK modulators. In this specification, symbols are generally referenced as being transmitted from BBIC to RFIC.

Sample: In the DigRF specification, a sample is a portion of a symbol comprising a specific number of bits. Samples are only referred to in receive mode. In one example, there are two samples per symbol, and 96 bits per sample (there are 64 bits of padding after each 16-bit I and Q), giving 192 bits per symbol. In our example, we are using an Rx signal with one sample per symbol and 96 bits per sample (still with 64 bits of padding), giving a total of only 96 bits per symbol. Samples are referenced (in the DigRF 1.12 specification) as being transmitted from the RFIC to the BBIC. Samples are multiplexed, meaning that the samples (or bits) within each symbol can be sent in any particular order – there is no requirement to send the first sample or bit of any symbol first, and there is no constraint on the RFIC to send full symbols at a time. Symbols sent from the BBIC to the RFIC are not multiplexed, and they are sent in the order needed for transmission over the air.

Signal integrity measurements

A logic analyzer complements an oscilloscope for making signal integrity measurements. A logic analyzer's timing mode is used to see the timing relationships between the signals and buses in your system, and the state mode is used to see a sequence of events. You can also use an Agilent logic analyzer's eye scan feature to simultaneously display eye diagrams on all channels, allowing you to identify problem signals for further analysis with an oscilloscope. For detailed parametric signal analysis, however, use an oscilloscope to look at things like overshoot, undershoot and jitter. Extensive literature is available on making signal integrity measurements with oscilloscopes.

The B4602A Signal Extractor Tool

For our example, we will look at data transmission from the RFIC to the BBIC with one sample per symbol. Each sample contains 16 bits I, followed by 16 bits Q, then 64 bits of padding.

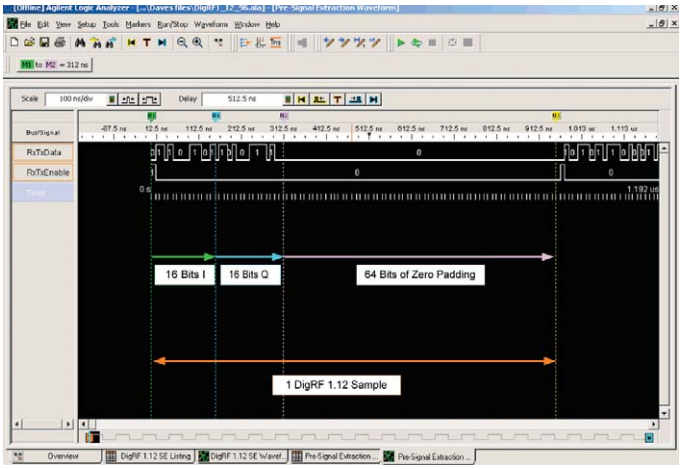


Figure 7. One DigRF 1.12 RxTxData sample viewed on Agilent's logic analyzer software

A logic analyzer can show us the samples, symbols and timing between them. It can also show the control data and clocks and strobe. However, when we need to demodulate, another step is needed. The VSA software doesn't need control data or padding, but it does need the IQ data and it needs it in parallel form. To get this data into the VSA, we'll need a new tool, the B4602A Signal Extractor tool.

The B4602A Signal Extractor Tool

Extracting the data

The Signal Extractor uses algorithms to extract the bits we need. Some algorithms come standard, and others you will need to create yourself. A useful way to think of the Signal Extractor tool is as a data extractor and logical reformatter. Both the DigRF 1.12 and the DigRF Version 3 specifications have IQ data that is interspersed with other data that the VSA doesn't need. Therefore, we must use the Signal Extractor to pull the IQ data out from the serial data stream and format it for the 89601A software.

The logic analyzer makes a data acquisition on the DUT. Once that information is in sample memory, the Signal Extractor pulls out the I and Q data and reformats it in whatever way we have defined. The extraction algorithms define what data to extract and how to label the data it extracts. For instance, in our example, the input signal is "RxTxData"(see Figure 8) Once we've extracted the data with the algorithm, there are two new signals (or signal buses in logic terms) labeled "My bus I" and "My bus Q." Depending on how the algorithm is defined, we could have a myriad of various signals to look at. The extraction algorithm is based upon a series of XML ASCII commands which are defined in a single algorithm file. You are free to choose your extraction algorithm or to customize the extraction algorithm to meet your needs. All algorithms are written in XML.

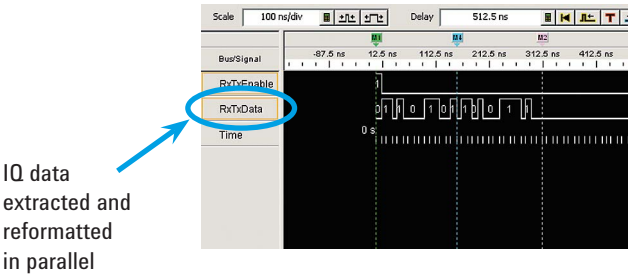


Figure 8. Original data, prior to using Signal Extractor

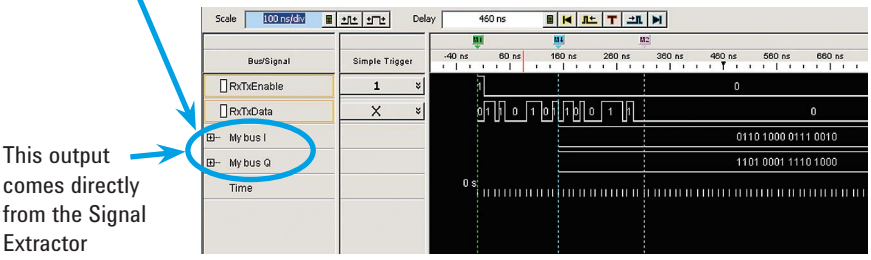


Figure 9. Signal Extractor has added two new signals, My bus I and My bus Q

The B4602A Signal Extractor Tool

Understanding the algorithm

The example algorithm we will use from the Signal Extractor is the "DigRF 1.12 Trigger Qualified" algorithm. This algorithm specifies the extraction to happen at a trigger point which the user sets to start at the first bit of our 96-bit frame (DigRF sample). The algorithm specifies the bits to extract once the trigger is seen by the logic analyzer.

To make things a little more obvious, we'll go step by step through the algorithm we will use on our DigRF 1.12 example.

This particular algorithm is built in to Signal Extractor. It's called "DigRF_Trig_Qual." The title refers to the fact that it's meant to execute with a simple trigger. In our case, that trigger is the RxTxEnable signal (see Figure 6).

Note that ours is not a true RxTxEnable signal, but one we created to make it possible to use a simple trigger. A true DigRF 1.12 RxTxEnable signal would most likely stay in a high state for the transmission of an entire DigRF sample.

Figure 10 is the "DigRF 1.12 Trigger Qualified" algorithm as we would see it if we were to double click its title in its folder.

```
- <ExtractorGrammar AlgorithmDescription="DigRF 1.12 Trigger Qualified">
    <Comment Value="DigRF 1.12 Trigger Qualified approach This version extracts I and Q values from
      the specified serial input. It depends upon the user to set up a trigger so that the starting
      sample is the first bit of the 96-bit frame. The user can use the logic analyzer trigger to specify
      this. In addition, the user should qualify the storage of the analyzer so that only when the
      RxTxEnable is true is data being stored. Finally, the user should also specify the starting
      sample in the signal extractor to be sample 0. Define a label that contains RxTxData (bit-0)." />
  - <ExtractorLabels>
      <ExtractorLabel Name="My bus I" Width="16" DefaultBase="Hex" VSAOutput="T"
        VSACompressionFactor="-96" />
      <ExtractorLabel Name="My bus Q" Width="16" DefaultBase="Hex" VSAOutput="T"
        VSACompressionFactor="-96" />
    </ExtractorLabels>
  - <ExtractorSequences>
    - <ExtractorSequence>
      - <ExtractorPatterns>
          <ExtractorPattern Value="bX" Width="1" Enabled="T" />
        </ExtractorPatterns>
      - <ExtractorCmds>
          <ExtractorCmd Cmd="LoadRange" BitStart="0" BitEnd="15" />
          <ExtractorCmd Cmd="WriteLabelTime" Name="My bus I" BitTime="15" />
          <ExtractorCmd Cmd="LoadRange" BitStart="16" BitEnd="31" />
          <ExtractorCmd Cmd="WriteLabel" Name="My bus Q" />
          <ExtractorCmd Cmd="GoTo" Bit="95" />
          <ExtractorCmd Cmd="JumpDone" />
        </ExtractorCmds>
      </ExtractorSequence>
    </ExtractorSequences>
  </ExtractorGrammar>
```

Figure 10. Signal Extractor "DigRF 1.12 Trigger Qualified" algorithm

The B4602A Signal Extractor Tool

Understanding the algorithm *(continued)*

The XML code may look a little daunting to a non-programmer, but it's fairly easy to follow. Let's break it up a bit:

Part 1: Extractor Labels

```
- <ExtractorGrammar AlgorithmDescription="DigRF 1.12 Trigger Qualified">
    <Comment Value="DigRF 1.12 Trigger Qualified approach This version extracts I and Q values from
      the specified serial input. It depends upon the user to set up a trigger so that the starting
      sample is the first bit of the 96-bit frame. The user can use the logic analyzer trigger to specify
      this. In addition, the user should qualify the storage of the analyzer so that only when the
      RxTxEnable is true is data being stored. Finally, the user should also specify the starting
      sample in the signal extractor to be sample 0. Define a label that contains RxTxData (bit-0)." />
  - <ExtractorLabels>
      <ExtractorLabel Name="My bus I" Width="16" DefaultBase="Hex" VSAOutput="T"
        VSACompressionFactor="-96" />
      <ExtractorLabel Name="My bus Q" Width="16" DefaultBase="Hex" VSAOutput="T"
        VSACompressionFactor="-96" />
    </ExtractorLabels>
```

Figure 11. Part 1 of the Signal Extractor algorithm "DigRF 1.12 Trigger Qualified"

The ExtractorLabels element contains output bus/signal and bus/signal folder definitions. Note that the "parent" (as the module name is often called) is ExtractorLabels and the elements listed under this are the "children." In other words, the children are subordinate to the parent (see Figure 12).

The B4602A Signal Extractor Tool

Understanding the algorithm *(continued)*

The following table from the Signal Extractor Help files shows the different input/output signal definitions associated with the "ExtractorLabels" children.

Table 1. IO signal definitions for ExtractorLabels

Name	Description
DefaultBase	Binary, hex, octal, decimal, signed decimal *In this algorithm, we've used "Hex."*
Name	'string' – Text name of output bus/signal to display *Our bus/label names are "My bus I" and "My bus Q." We can see this in Figure 11.*
VSACompressionFactor	'number' – Optional value used to tell the Agilent 89601A vector signal analysis software about the serial compression or expansion that is occurring between the input data to the output data. A value of –16 implies that for every 16 input states the output bus/signal has 1 state. A value of +4 implies that for every 1 input state we are generating 4 output states. *In our case, we already know that our signal has 16 bits of I, 16 bits of Q (don't forget the order!) 64 bits of zero padding, for a total of 96 bits. There is a negative sign in front of the value because we are kind of compressing the data. The VSACompressionFactor is –96 in this algorithm.*
VSAOutput	'F' – The bus/signal cannot be used with the Agilent 89601A vector signal analysis software. 'T' – The bus/signal can be used with the Agilent 89601A vector signal analysis software. *Of course for our purposes, we do want to use it with the 89601A software!*
Width	'number' – Width of the output bus/signal in channels (1 to 128) *We are telling it to make the bus width 16 bits, meaning it will be a parallel bus 16 bits wide. If we wished to output serial data, then the bus width would be 1 bit wide.*

The B4602A Signal Extractor Tool

Understanding the algorithm *(continued)*

Part 2: Creating the sequence of events
Here is where we really create the algorithm. Before coding the XML, it's wise to create a flow chart for the algorithm.

XML and Signal Extractor have certain conventions. First we must define the ExtractorSequences, which tells the Signal Extractor to expect one or more ExtractorSequence followed by a series of commands (ExtractorCmds) that will define the exact bits to extract and tell the Signal Extractor where to put them.

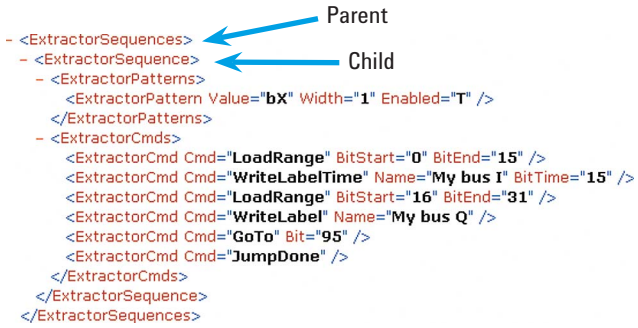


Figure 12. Part 2 of the Signal Extractor algorithm "DigRF 1.12 Trigger Qualified"

ExtractorSequences is the parent here.

Within the ExtractorSequences parents, we define a similar label, ExtractorSequence. We must do this for each and every separate sequence we wish the Signal Extractor to follow.

The B4602A Signal Extractor Tool

Understanding the algorithm *(continued)*

As XML and Signal Extractor conventions dictate, we must also define an ExtractorPatterns label. This label tells the logic analyzer what pin mapping to expect from the logic analyzer probe.

This table from the Signal Extractor help files explains elements associated with the ExtractorPatterns label.

Table 2. Elements associated with the extractor patterns

Name	Description
Enabled	'F' – Indicates this pattern is initially disabled 'T' – Indicates this pattern is initially enabled *We're generally going to have this set to "T."*
Value	'binary_or_hex_number' – Binary or hex pattern to search for first. Use a leading 'b' or 'h' and 'X' for don't care on some bits. *In our algorithm, we've set it to Binary as the leading bit and don't care (X) for anything else – "bX."*
Width	'number' – Width of the pattern that you are looking for (1 to 128) *Now we are referring to the actual serial DigRF 1.12 data stream, which is, of course, "1" bit wide.*

The B4602A Signal Extractor Tool

Understanding the algorithm
(continued)

Part 3: Extractor commands

```
- <ExtractorCmds>
    <ExtractorCmd Cmd="LoadRange" BitStart="0" BitEnd="15" />
    <ExtractorCmd Cmd="WriteLabelTime" Name="My bus I" BitTime="15" />
    <ExtractorCmd Cmd="LoadRange" BitStart="16" BitEnd="31" />
    <ExtractorCmd Cmd="WriteLabel" Name="My bus Q" />
    <ExtractorCmd Cmd="GoTo" Bit="95" />
    <ExtractorCmd Cmd="JumpDone" />
```

Figure 13. Part 3 of the Signal Extractor algorithm "DigRF 1.12 Trigger Qualified"

This is where we tell the Signal Extractor exactly what data to pull and where to put it. The ExtractorCmd element listing in the Signal Extractor help files lists all the possibilities. There are far too many to list here!

The term LoadRange is used to tell the Signal Extractor to grab a range of bit values starting at bit 0 (BitStart="0") and ending at bit 15 (BitEnd="15"). Note that we could, if we wish, tell it to load a single bit (LoadOne) or a specific bit ('Load' Bit='0') – see the Signal Extractor help files for more information.

"WriteLabelTime" writes the values extracted in the first line to the specified bus/signal (Name='string' or "My bus I" in our case) and then copies the time associated with each bit.

We repeat both commands for the Q values we want to extract.

The next line tells the Signal Extractor to jump to bit 95, ignoring the 64 bits of padding. It's important to remember that there is such a thing as a bit "0"and this is where the logic analyzer starts counting. Thus, bit 96 will start the entire pattern over again.

Specifically, "Jumpdone" tells the Signal Extractor tool to jump to the end of the command section and return to the pattern matching (ExtractorPatterns) section of the algorithm.

The B4602A Signal Extractor Tool

Understanding the algorithm
(continued)

Part 4: Modifying an existing algorithm
Modification is quite simple. Open the algorithm in Notepad or WordPad (make sure you tell it to open "All files," as neither tool will recognize the XML file extension). Before proceeding any further, save it with another title. All values are text strings in XML. In a line of code, for example:

<ExtractorLabel Name='My bus I' Width='16' DefaultBase='Hex' VSAOutput='T' VSACompressionFactor='–96'/>

we could change the ExtractorLabel name to 'My serial data' or some other name. We must adhere to XML and Signal Extractor conventions and elements, but any text string can be modified.

We might change these lines of code:

<ExtractorLabel Name='My bus I' Width='16' DefaultBase='Hex' VSAOutput='T' VSACompressionFactor='–96'/>
<ExtractorLabel Name='My bus Q' Width='16' DefaultBase='Hex' VSAOutput='T' VSACompressionFactor='–96'/>

to this:

<ExtractorLabel Name='My parallel bus I' Width='32' DefaultBase='Binary' VSAOutput='T' VSACompressionFactor='–128'/>
<ExtractorLabel Name='My bus Q' Width='32' DefaultBase='Binary' VSAOutput='T' VSACompressionFactor='–128'/>

if we had a serial input bus that had 32 bits of I, followed by 32 bits of Q data and 64 bits of padding, and we wanted the Signal Extractor buses to show up in binary format (note that the format can be changed in the logic analyzer waveform screen). *Keep in mind that if you change the name in one part of the algorithm, you must change it in all places it's referred to.*

The B4602A Signal Extractor Tool

Understanding the algorithm *(continued)*

Once you've made the changes you want to the algorithm, save as type "All files" with the .xml extension left on. The new algorithm will look like this:

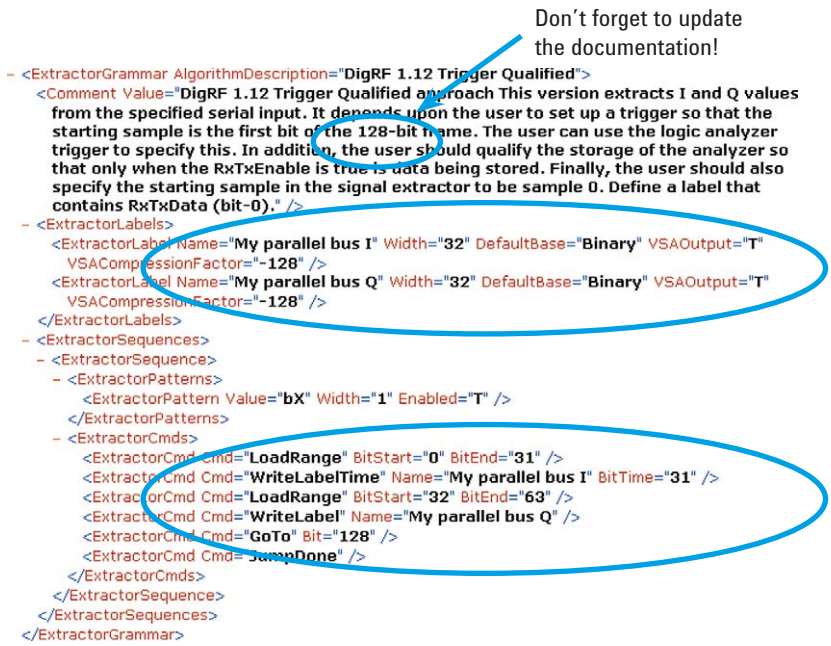


Figure 14. Modified algorithm

The logic analyzer's waveform view will look like this:

The two buses can be expanded to show the full 32 bits for each.

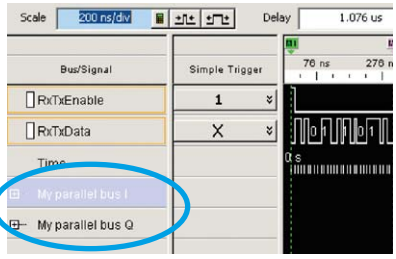


Figure 15. Bus name modifications

Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Getting started

Once both the logic analyzer software and 89601A VSA software are installed, we need to get them speaking to each other. Go to **Start > Programs > Agilent 89600 VSA > Logic Analyzer > IO Connections**.

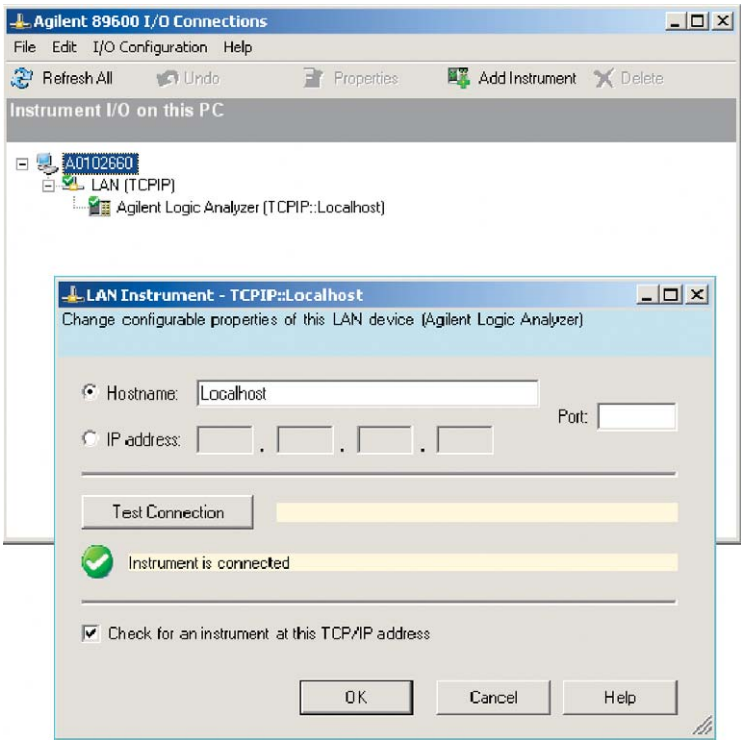


Figure 16. Logic analyser/VSA communication setup

Right click **LAN(TCPIP)** and then select **Agilent Logic Analyzer**, and click **OK**.

In the Hostname box, type "Localhost," and click **Test Connection**. You should get a green check mark as shown in Figure 16. The VSA and logic analyzer software are now connected. The VSA uses the VISA Assistant in the IO Libraries Suite to "speak" with the logic analyzer software. Thus we need to give the logic analyzer a "name" for the VSA to speak to.

Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Logic analyzer

Setting up the logic analyzer

When we launch the logic analyzer software, we will need to set it up for a capture. The first thing we need to do is set up our Overview window. The next thing we'll do is setup bus, thresholds, and signals. Ensure that the logic analyzer cables are connected to the probe properly, and that the probe is connected to the DUT. Supply power to the DUT and power on the logic analyzer. The outer end connectors on logic analyzer cables are referred to as "pods" and are numbered Pod 1, Pod 2, Pod 3 etc. The number of pods for each logic analyzer module (card) depends on the card itself. Generally there are two. For our DigRF 1.12 Signal, we will need only one Pod. Pods generally contain 40 pins, meaning there are 38 data channels and two clock channels per pod. The probe being used must have the same number of channels. One side attaches to the logic analyzer pod, and the other portion (the probe "tip") attaches to the DUT.

The Overview window and the Signal Extractor tool

The Overview window is typically the first window that opens up. The default setting has one graphical user interface (GUI) for the default logic analyzer (the name of the logic analyzer module will be whatever hardware the software finds), and there will be a list and waveform window attached to that module.

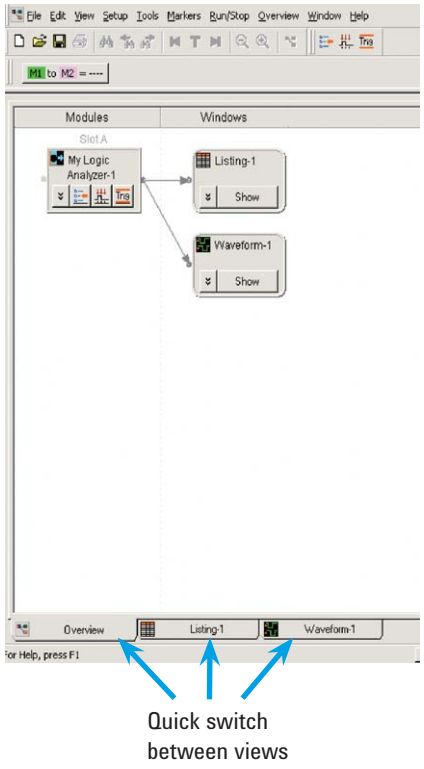


Figure 17. Default configuration

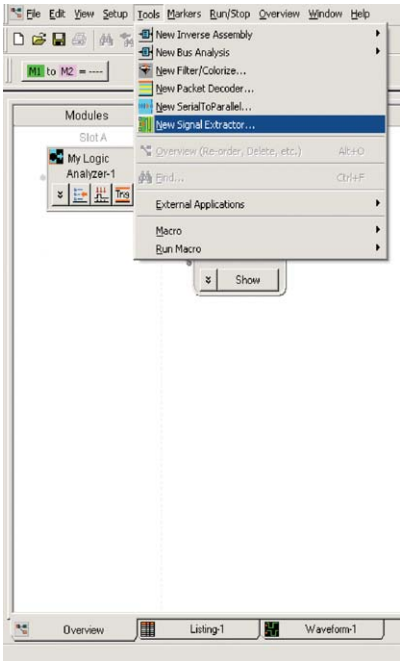


Figure 18. Adding a Signal

Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Logic analyzer *(continued)*

Extractor tool

To hook up the Signal Extractor tool, go to **Tools > New Signal Extractor** and the software will automatically place a Signal Extractor tool into the path between the logic analyzer module and the list and waveform windows.

To call up the algorithm, click **Properties** of the Signal Extractor GUI, then select **Load Algorithm**. This gives us an option to browse the Signal Extractor files and pick our algorithm. The one we're using here is the same we looked at in the previous section.

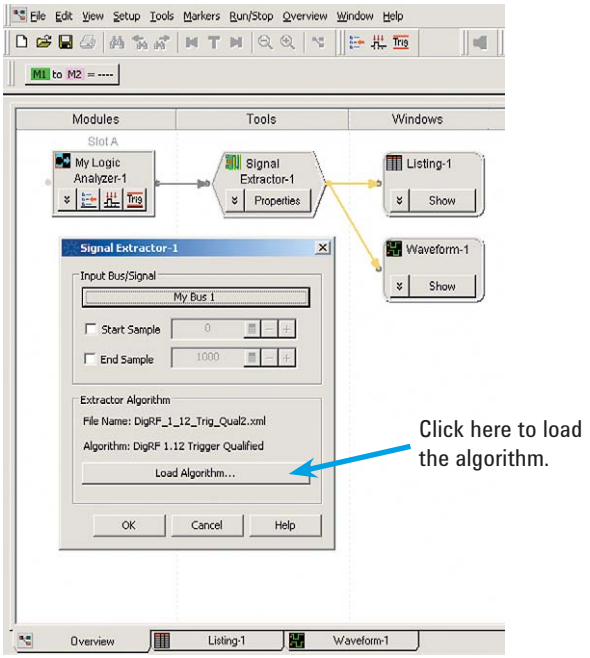


Figure 19. Loading the algorithm

Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Logic analyzer *(continued)*

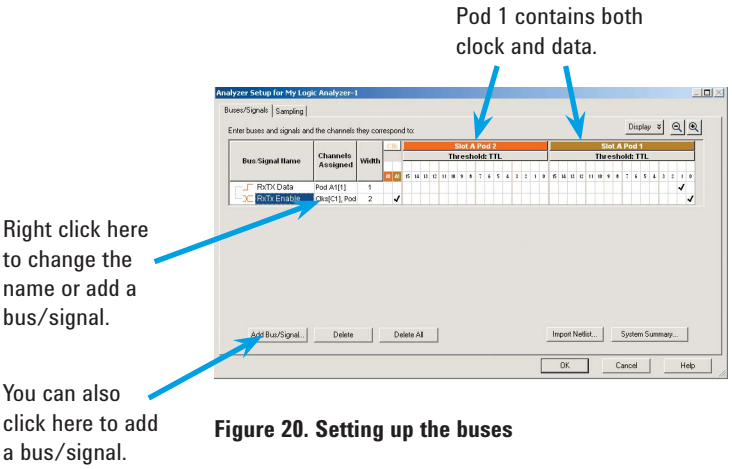


Figure 20. Setting up the buses

Setting up the bus, thresholds and signals

To set up the bus, thresholds, and signals go to **Setup > Timing/State** (see Figure 21). On the sampling tab, select **State**, and set the trigger to 99% post store. We'd like a little bit of pre-trigger information but we want the bulk of the information after the trigger.

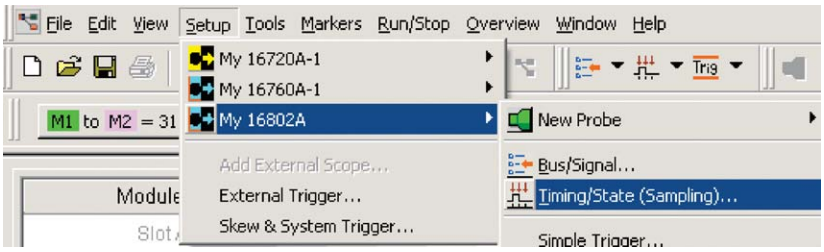


Figure 21. Setup > Timing/State

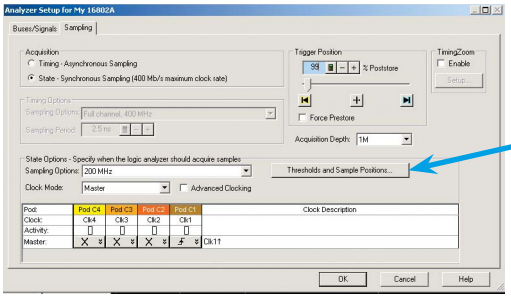
Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Logic analyzer *(continued)*

Memory depth

The acquisition depth we'll leave at 1 M, as that is more than adequate for our needs. In logic analyzer terms, that's 1 MSample of memory. We can collect 1 Mbit of data. In our case, since we know that the DigRF samples will be only 96 bits long, we have plenty of memory.

Under **Sampling Options** (as shown in Figure 22) there will be a couple of choices, depending on the logic analyzer you're using. We leave our sample speed at 200 MHz. State mode sample speed matches your DUT's clock rate, up to certain rates. Rule of thumb suggests that we pick the lowest sample rate that will meet our needs. We could switch it to a higher rate, but there's simply no need to and there are other tradeoffs for higher sampling capability. We leave the clock as master in this case because we're only using one logic analyzer module (several modules are needed for high channel count on parallel bus signals and the master clock can control the timing for all).



To use eye finder, click this button.

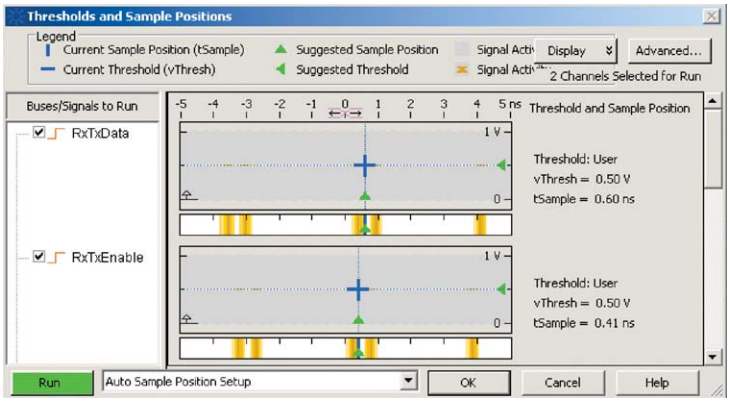
Figure 22. Sampling setup window

Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Logic analyzer *(continued)*

Thresholds and sample position

Once the logic analyzer is correctly hooked up to the DUT and powered on, activity on each active pod will show up in the bus/signal display under the correct pod. When hooking up the logic analyzer, there will be specific voltage thresholds we want it to look at (meaning where we will sample on the rising or falling edge of the bit). We also want to optimize the position on the bit where the logic analyzer takes the sample. Ideally, we want the sample taken directly in the middle of the bit. There is a tool built into the logic analyzer specifically for this called "eye finder" (Figure 23). In state mode, the logic analyzer looks at signals from the device under test, figures out the threshold voltage that results in the widest possible data valid window, then figures out the location of the data valid window in relation to the sampling clock and automatically sets the threshold voltage and sampling position. The term "data valid" simply means when the data is either high or low, not transitioning (see Figure 24). To use eye finder, we select **Threshold and Sample Position**. For our DigRF 1.12 signal, the VIH is 0.7 (see Appendix A under Voltage Levels) so we will want to set our threshold to about 0.5 volts to ensure that the logic analyzer properly identifies a logic one. The Threshold and Sample Position tool should also see this and set it accordingly. Alternately, we can set the threshold values manually under the **Buses/Signals** tab (Figure 25).



Eye finder shows the correct threshold, and the proper sample position.

Figure 23. Eye finder

Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Logic analyzer *(continued)*

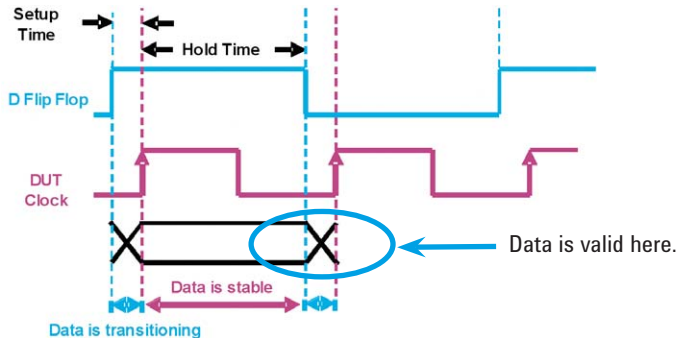


Figure 24. Data valid

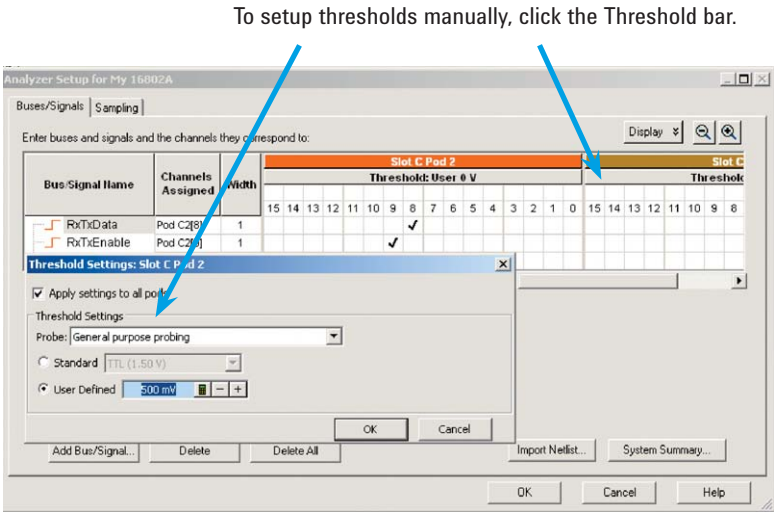


Figure 25. Manual threshold setup window

Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Logic analyzer *(continued)*

Trigger setup

Our example uses a very simple trigger in the form of an event output. On a DigRF 1.12 signal you will generally want to use the RxTxEnable signal.

We access the trigger menu either by going to the **Setup** menu in the tool bar, or going back to the Module GUI (Figure 26) in the Overview window and selecting the Trigger button. As we can see in Figure 27, we've put in very simple trigger that tells the logic analyzer to fill memory as soon as it sees the RxTxEnable signal go high. Since we've put in 99% post store, we'll capture 1% prior to the actual trigger event so we can see a few events leading up to the trigger point.

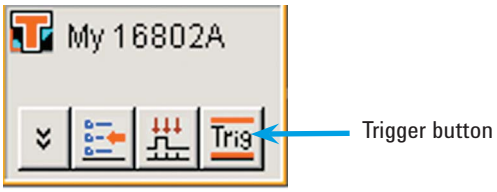


Figure 26. Logic analyzer model GUI

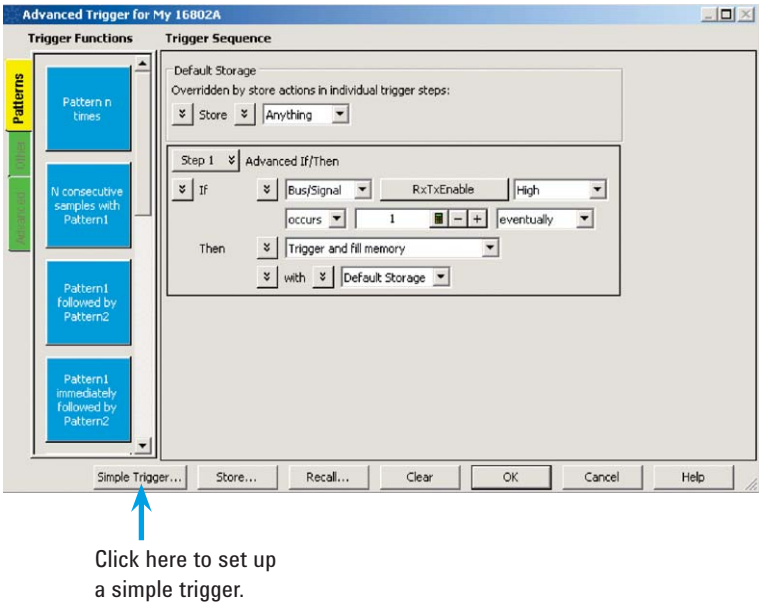


Figure 27. Trigger setup

Once we're ready, we can run the logic analyzer. On the tool bar, there is a green arrow, and we simply click on that to run the instrument.

Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Logic analyzer *(continued)*

Logic analyzer results

We can easily view the data and the waveforms by switching between windows. In our example, remember we have a 96-bit sample, where there are 16 bits of I, followed by 16 bits of Q and 64 bits of padded zeros.

Viewing the data

We can view the data in List view or Waveform view. In fact we can view it before and after we use the Signal Extractor. Right clicking the arrow between the List view and the Signal Extractor GUIs in the overview window allows us to add a List view window (List view must be dedicated to serial or parallel data, not both. Since the newly extracted data is parallel, we'll need two List view windows).

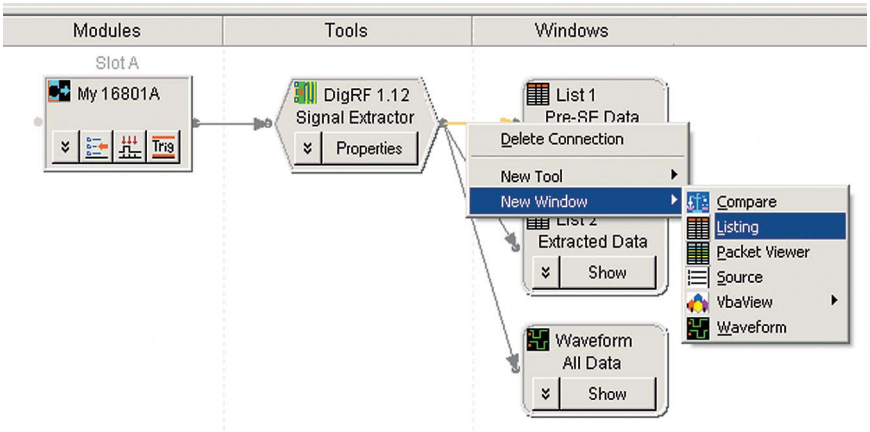


Figure 28. Adding a window

Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Logic analyzer *(continued)*

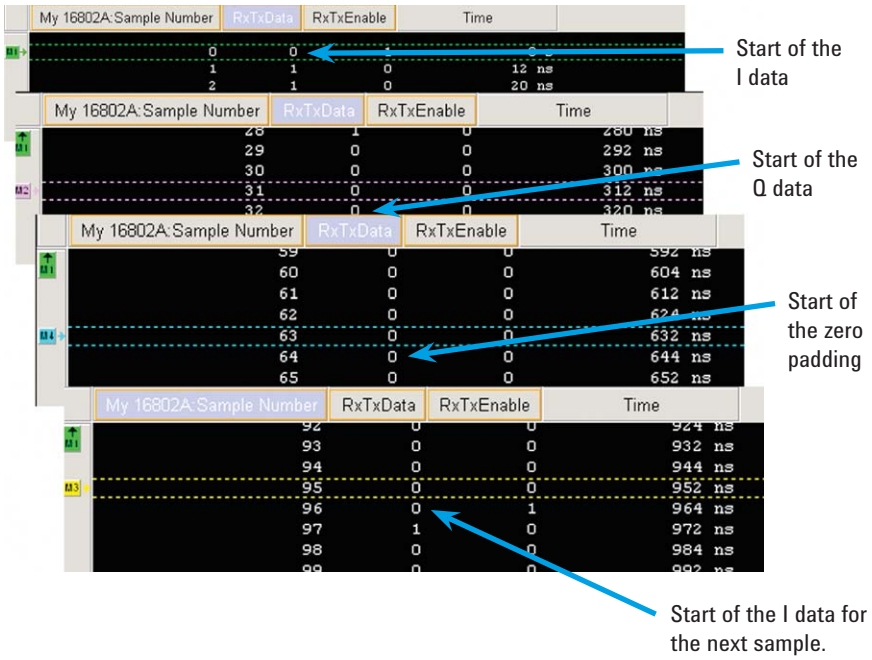


Figure 29. List view of pre-extracted data (serial). We can scroll down and count the bits using markers.

Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Logic analyzer *(continued)*

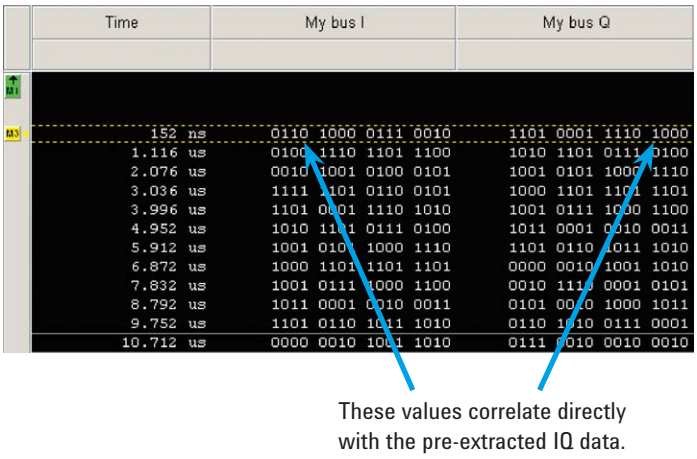


Figure 30. List view of DigRF 1.12 data after signal extraction

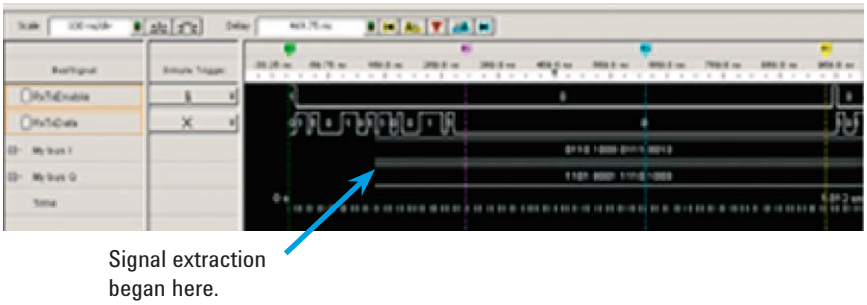


Figure 31. Waveform view of data

Figure 32. Waveform view in chart mode. In this mode, all the symbols are displayed as if they are in the time domain.

Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Vector signal analyzer

Setting up the vector signal analyzer

Once we start up the VSA software, we must first set up the digital input properties. We access this from the Input menu on the toolbar, as shown in Figure 33. Figure 34 explains how to set up these properties.

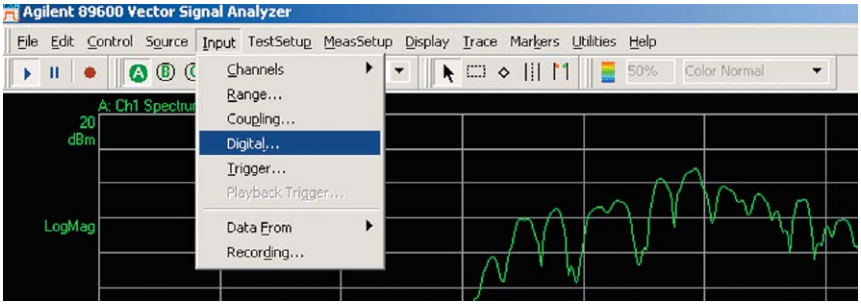
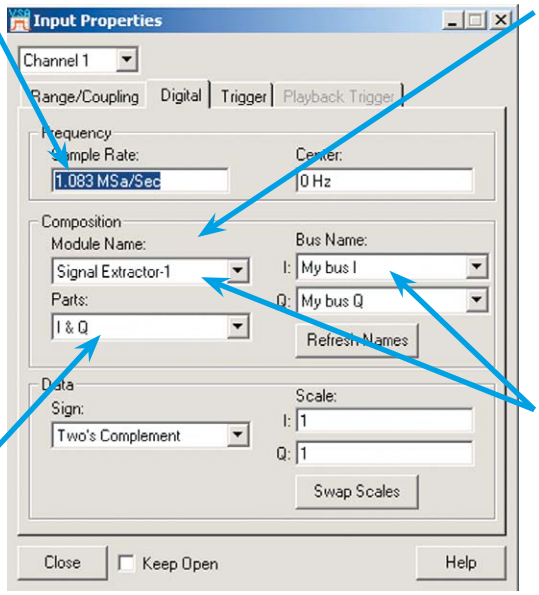


Figure 33. Drop-down menu for digital input properties

Set the GSM sample rate here. This sample rate must be correct or the signal will not demodulate properly. The VSA sets its span according to the sample rate set here. GSM is 4X oversampled, so our sample rate is 1.0833 MSa/S.

The Parts pull-down menu must be set to I&Q.



The module name is the Signal Extractor! The VSA takes the IQ data only from the Signal Extractor after the IQ data has been reformatted.

All the names will match the logic analyzer names.

Figure 34. Digital Input Properties dialog box

Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Vector signal analyzer *(continued)*

The Digital Input Properties dialog box is the most important setup window in the VSA software. This is where we tell the VSA what to expect. We set the sample rate, module name, bus names, and parts label. We can also chose the data types – either two's complement or offset binary – and we can choose to swap the scales. This example leaves both these at the default. Note that the center frequency is set to 0 Hz. This is because we're using IQ signals. In some cases, a digital radio may have a digital LO and mixer. If complex data is taken on the other side of the mixer, then it has a non-zero center frequency. We would then type this center frequency in the Digital Input Properties dialog box so the displays are properly labeled.

Next, it's always a good idea to check the measurement setup properties in the MeasSetup Properties dialog box. We can check the span, look at the spectrum to make sure it makes sense (the entire signal should be displayed. If it looks cut-off, the sample rate is likely to be incorrect).

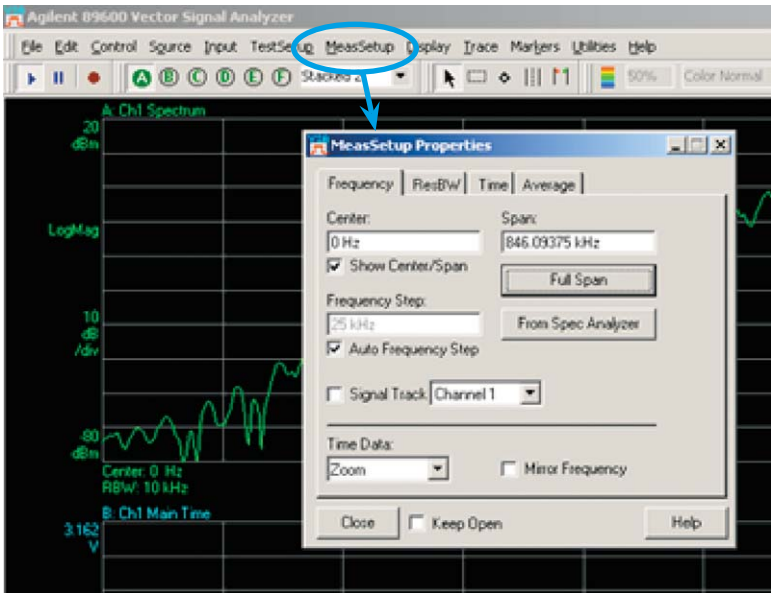


Figure 35. MeasSetup Properties dialog box

Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Vector signal analyzer *(continued)*

Now we are ready to set up the demodulation. We first open the Demod Dialog by pulling down the MeasSetup menu and selecting **Demodulation > Digital** (see Figure 36).

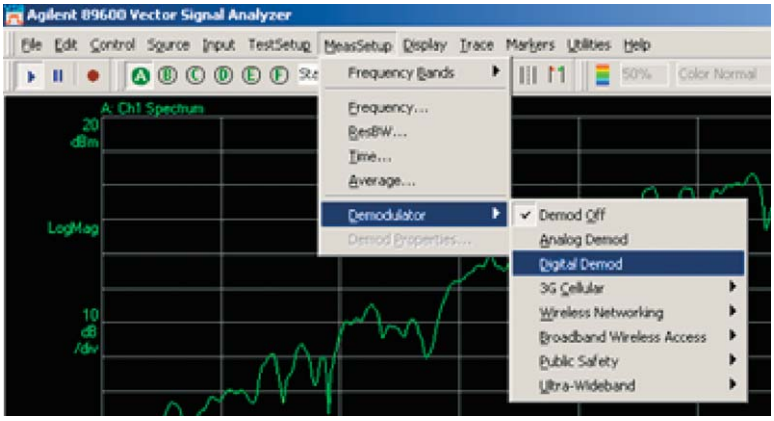


Figure 36. Demod drop-down window

Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Vector signal analyzer *(continued)*

Once we've selected that, we need to set up the demod properties.

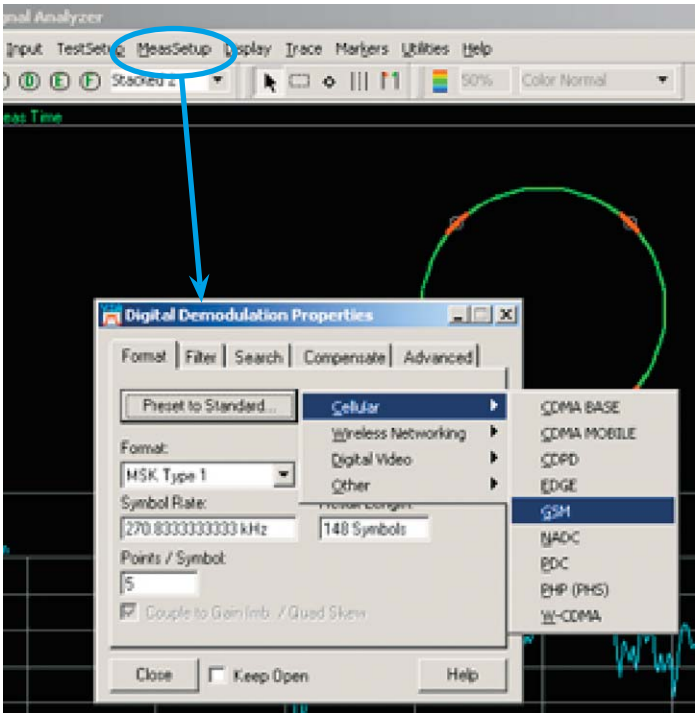


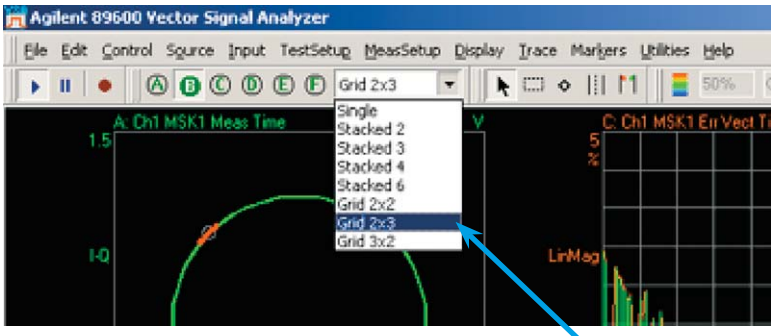
Figure 37. Selecting the modulation format

Our DigRF 1.12 example uses the GSM format. When we select this, the VSA software automatically sets up the correct parameters such as symbol rate (270.8333 kHz), modulation format (MSK Type 1) and filtering.

Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Vector signal analyzer *(continued)*

Once the modulation parameters are set up, we can customize the displays. Let's start by putting up a set of 6 displays. This is done by simply pulling down the Grid menu.



Simply pull the menu down and select the display grid you want to use.

Figure 38. Changing the number of displays

We can then modify the contents of the displays. There are quite a few choices! In our example, we'll take a look at the eye diagrams for the I and Q. On traces E and F, there is likely no display at default. So we right click the title and select **Edit**.

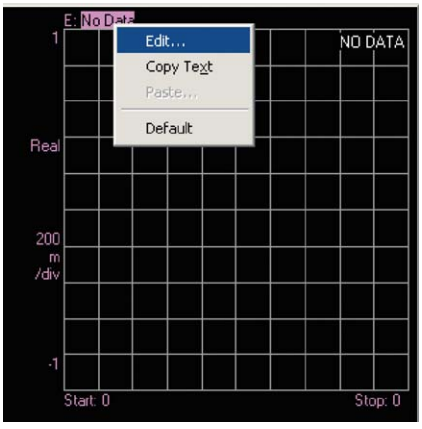


Figure 39. Customizing display contents: X-axis

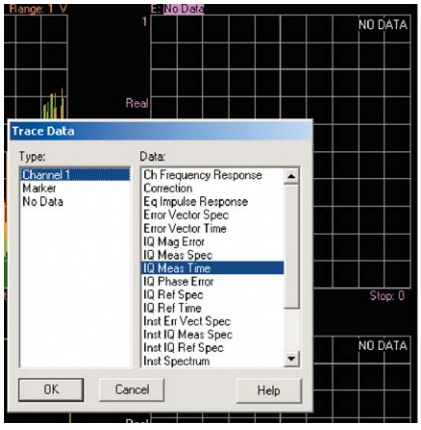


Figure 40. Selecting the display type for the X-axis

Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Vector signal analyzer *(continued)*

We have several choices. Since we'd like to look at the eyes, we're going to select **IQ Meas Time** for the X-axis and **I-eye** for the Y-axis.

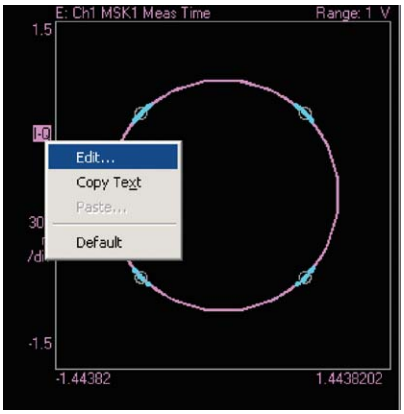


Figure 41. Changing the Y-axis

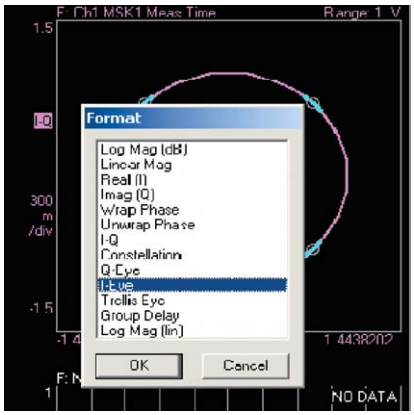


Figure 42. Selecting the format

We will do the same thing with display F, but this time we select the **Q-Eye**.

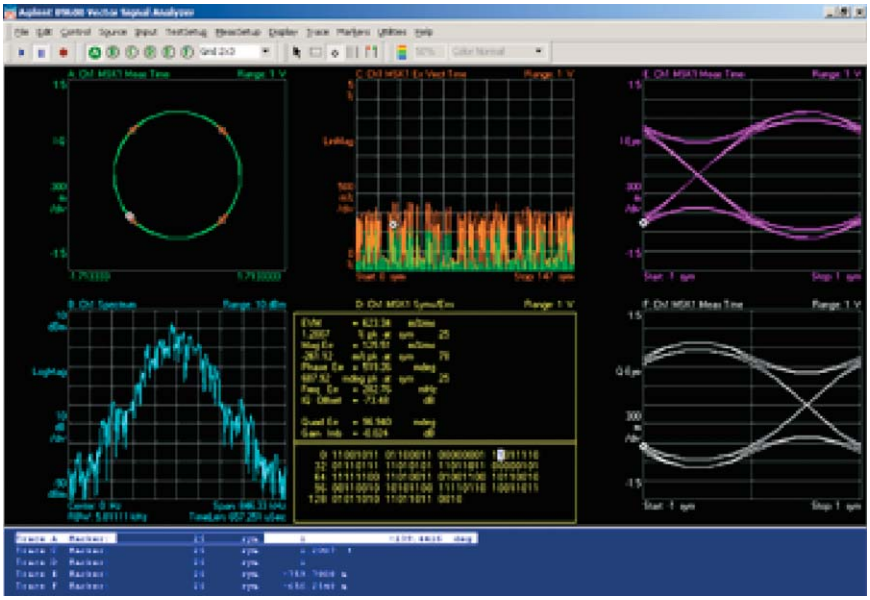


Figure 43. Full 2x3 set of displays

Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Vector signal analyzer *(continued)*

Vector signal analysis results

We've already seen the eye diagrams but let's examine a couple of the other displays we can see with the VSA software and take a look at what's presented.

Sync Search

One very nice feature is the Sync Search. We can take a pattern of data and tell the VSA to find it. In this case, the 01111100 represents the first four symbols of the DigRF data in the first 96-bit sample. We might like to use this feature to ensure that the BBIC is sending the appropriate data, for example. There are also several GSM and EDGE training sequence preset patterns to choose from.

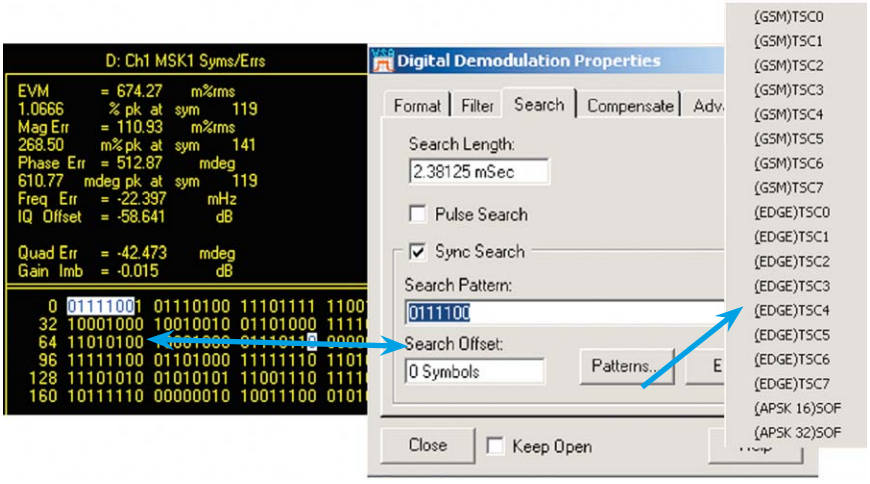


Figure 44. Sync Search helps you find specific data patterns.

Setting up the Logic Analyzer with Signal Extractor and the 89601A Software

Vector signal analyzer *(continued)*

Syms/Errs summary table and more

The Syms/Errs table is a very handy feature that shows us overall modulation quality. It will also show us peak values for metrics such as EVM, magnitude and phase errors. Marker coupling allows us to view a troublesome symbol in other domains, too, such as IQ phase and magnitude error.

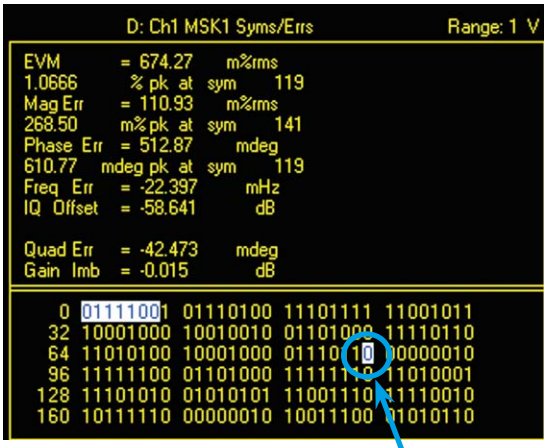


Figure 45. Syms/Err table

Marker coupling lets us look at a single symbols in different domains.

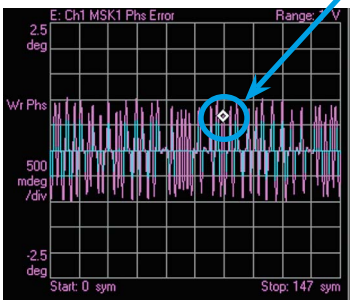


Figure 46. IQ phase error display

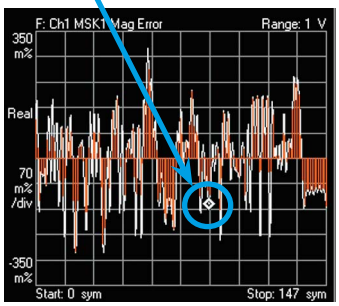


Figure 47. IQ magnitude error display

Summary

The transition of interfaces from analog to digital is a trend that is going to continue. The availability of the proper tools for test and measurement of such changing interfaces paves the road for a smooth transition. A seemingly complex task has been shown to be fairly simple using standard test and measurement equipment with the new Signal Extractor tool.

Appendix A: DigRF 1.12 Digital Physical Layer Overview

The DigRF 1.12 standard defines the digital interface between BBIC and RFIC for GSM, GPRS and Edge. Its purpose is to allow open communication between BBIC and RFIC regardless of the vendor of either, while reducing pin count and maintaining efficiency through the use of standardized digital (versus analog) interfaces.

The standard does not address the physical layer of GPRS. The physical layer is as defined in

[1] GSM 05.10, Radio Subsystem Synchronization (section 6.4 in version 7.1.0, or equivalent in later versions)

[2] GSM 05.04, Modulation (section 3.2 in version 8.0.0, or equivalent in later versions)

The standard itself does not define the TDMA protocol layer. These layers will be dependent on the requirements as stated in the protocol layer specifications for the same telecommunications standards.

In particular, the DigRF 1.12 standard addresses the 2.5G GSM (slot classes 1 through 12) on the logic, electrical and timing level. A separate standard, DigRF Version 3, will cover the 3G implementation of digital interfaces.

The latest version of the DigRF 1.12 specification can be found at **www.digrf.com**.

Direct conversion and near-zero IF

The DigRF 1.12 interface was designed specifically to support both direct conversion and near-zero IF radios. These terms may be familiar to RF engineers, but are not likely to be familiar to digital engineers.

The term “direct conversion” refers to a radio that uses IQ signals for modulation instead of the older analog modulation technology, resulting in an initial intermediate frequency at 0 Hz. “Direct conversion” is also called “ZeroIF” or “ZIF.” Figure 52 shows the spectrum of a QAM modulated signal at 0 Hz center frequency.

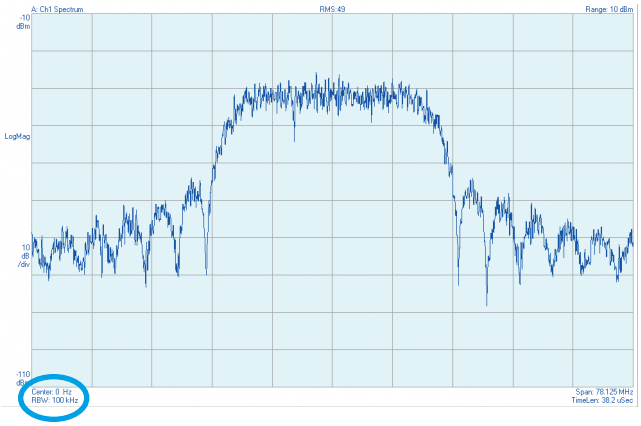


Figure 48. Spectrum of an IQ signal (QAM16)

Appendix A: DigRF 1.12 Digital Physical Layer Overview

The term “near-zero IF” or “NZIF” refers to a circuit where the IQ signals have been mixed up from DC to a low intermediate frequency. Typically this allows for gain to be distributed between the DC and IF stages, reducing the probability of DC offset, and requiring fewer ADCs. DC offset occurs when there is imbalance between the phase of the I and Q signals, in which causes a DC component that distorts the signal and limits ADC dynamic range.

Nomenclature conventions within the DigRF 1.12 specification

It is a good idea to separate out some of the nomenclature used in the DigRF 1.12 specification and make note of where these terms are used. See the discussion of “symbol” versus “sample” on page 11. Symbols sent from the BBIC to the RFIC are not multiplexed, and are sent in the order needed for transmission over the air.

“Transmit” versus “receive”

It is important to note that when data is being sent from the BBIC to the RFIC, this transfer is referred to as being transmitted. When data is sent from the RFIC to the BBIC, this transfer is referred to as being received. The term “transmission” here is used to refer to the passing of data between the RFIC and BBIC interface, in any direction.

The RFIC and BBIC

RFIC: The RFIC contains the GMSK and 8PSK modulators. It may or may not contain a data/bit buffer and it may or may not contain a digital filter. This has been left entirely up to the designer, and the choices are all supported by the BBIC.

BBIC: The BBIC does generally contain a data/bit buffer, and it may or may not contain a digital filter. If it does contain a digital filter, it must also provide a filter bypass. The digital filter typically must match the required analog filter in the RFIC. When the digital filter is implemented in the BBIC, it is generally a programmable filter so BBICs and RFICs from different vendors can be interchanged.

Both the BBIC and the RFIC must support, at a minimum, 16 bits per sample at 2 samples per symbol. The standard clock (SysClk) rate is 26 MHz. There is also an optional 52-MHz mode.

Appendix A: DigRF 1.12 Digital Physical Layer Overview

The DigRF 1.12 interface

The DigRF 1.12 interface requires 8 pins per ASIC.

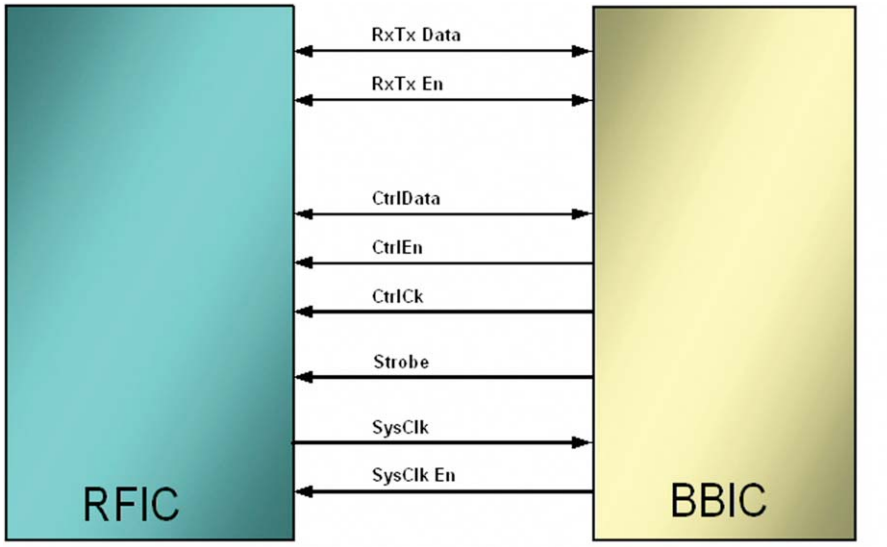


Figure 49. DigRF 1.12 ASIC interfaces

The RxTx interface

The RxTx interface is serial and bi-directional, and we can see on the first pin that the receive and transmit data must be multiplexed. The IQ samples (not symbols – here we are just talking about individual samples that comprise a symbol) may or may not be filtered, and come directly from in the BBIC. The SysClk (system clock) is used for clocking the data, and the RFIC is able to control the system clock via the SysClkEn line. The enable signal is used to tell either the RFIC or the BBIC that it is ready to transmit.

Control

The control interface is also serial and bi-directional, and in DigRF 1.12 it is on separate lines (In DigRF Version 3 the control data is multiplexed in with the IQ data on the same line.) The CtrlData is clocked using the control clock (which originates in the BBIC), and the control enable line is set by the BBIC, which is the master.

Strobe

The strobe is used as a time-accurate signal for over-the-air timing. It is used instead of the SysClk for extremely accurate timing of events. The strobe is driven by hardware within the BBIC.

Appendix A: DigRF 1.12 Digital Physical Layer Overview

Modes of operation

There are three modes of operation discussed in the DigRF 1.12 Specification. The two transmit modes take into account the optional existence of a data buffer within the RFIC. Transmit Block mode is used for data transmission from BBIC to RFIC when a buffer is present in the BBIC. Transmit Stream mode is used when the RFIC does not contain a buffer. Receive mode is used for transmission from RFIC to BBIC. In Transmit mode (symbols are sent from BBIC to RFIC), the symbols are sent in bursts, and each symbol is represented by four bits in order (not multiplexed). In Receive mode, where the bits are being sent from the RFIC to the BBIC, the IQ samples are multiplexed.

1. **Transmit Stream mode**

   In Transmit Stream mode, the RFIC does not contain a buffer, so the RFIC receives the symbols at the true symbol rate for modulation. There are four data bits per symbol and these are transmitted from the BBIC at about 1.083 Mbps (SysClk/24). The symbol rate is equal to the bit rate/bits per symbol and is approximately 270.833 kbps, as determined by the GSM specification. Do not confuse the system clock rate with the symbol or bit rate.

   Transmission of these bits continues until the buffer in the BBIC is emptied. Note that the BBIC is required to transmit whole symbols only, and the RFIC may require a pre- or post-amble in addition to the data (these pre/post-ambles may not exceed 32 bits). RFICs are required to specify and provide a constant group delay from the assertion of RxTxEn to the transmit chain outputs.

2. **Transmit Block mode**

   In this case, the bits are transmitted directly at a rate of 26 MHz (versus 26/24 MHz), which is a much higher rate. However, since they will be contained in a buffer within the RFIC prior to modulation, the output symbol rate (over the air) will be the same 270.8333 kbps (as set by the GSM standard). As in Stream Mode, there are 4-bit symbol bursts, but the transmission may be discontinuous. For example, an entire slot may not be contiguous as sent from the BBIC to the RFIC. In this case, there are requirements placed on the length of non-transmission: the gap in transmission must last a minimum of two system clock cycles (2/SysClk or about 760 nS). The RxTxEn signal must also be placed in the same way as if the burst difference were quite long (meaning that if the RxTxEn is high during transmission, it should be put low during the gap).

   The bits are sent from the BBIC to be held in the RFIC buffer for modulation. Also, in block mode, the RFIC must be told how many symbols will be sent from the BBIC. This information can be sent in several ways: via the control interface, events on the strobe signal, buffer contents, counts, etc. A constant delay between the assertion of the strobe signal and the actual transmission in block mode is determined by the RFIC and is provided to the BBIC.

Appendix A: DigRF 1.12 Digital Physical Layer Overview

3. Receive mode (Rx mode)

Receive Mode is used when the RFIC is receiving the bits down from the antenna and forwarding them to the BBIC. It's a more complicated than the first two modes, and the sample format is highly dependent on RFIC design. RxTxData to the BBIC contains multiplexed data and there are several possibilities for number of I, Q and padding bits per sample and the number of samples per symbol. The BBIC must accommodate these variables, which are set by the RFIC into the BBIC via control signals. The RFIC determines sample order (I first or Q first).

Bits per sample:
The number of bits per I and Q sample can be set to any integer value from 1 to 24. This is done to accommodate higher oversample ratios and dynamic range when such electronics become available.

If the RFIC contains a digital filter (it is optional), then the output of the digital filter is typically 16 bits (this conforms to a DSP word size). However, all 16 bits are not always significant. As we've seen in the Transmit modes, there are only 4 bits per symbol, so it is not necessary to use all 16 bits output from the filter.

When the RFIC is designed/programmed, it is up to the designer to ensure that the number of bits per sample set is appropriate to the RFIC or the data will be meaningless.

Padding bits are used when the I and Q samples will not occupy every system clock cycle. The number of pad bits is also set by the RFIC, and it will be in the range of 0 to 64 bits per sample.

One suggestion in the DigRF standard is one or two samples per symbol, with each sample containing 16 bits of I, 16 bits of Q, and 64 bits of zero padding. At 1 sample per symbol, we would have 96 bits per symbol, and this should be sufficient to occupy each clock cycle.

Voltage level requirements

The voltage level requirements are important because we must know what to set the logic analyzer threshold to in order to capture the correct logic levels. Using Table 7 in the DigRF 1.12 spec, we can see that ViL gives us a range from –0.3 to 0.3*VCC (the –0.3 V is to account for undershoot), and that ViH has a range of from 0.7 VCC to VCC +0.3 (the +0.3 V is to account for overshoot). Thus for a logic high, we will set our logic analyzer threshold to 0.5 V, which is below the lower threshold for ViH. Anything lower than that will be treated as a low by the logic analyzer.

Glossary

ASCII American standard code for information interchange: ASCII is the binary code for representing characters in computers.

ADC Analog-to-digital converter: A circuit that converts analog signals to digital. ADCs typically have a resolution of many bits. For instance, a single analog sinusoidal may be represented by 16 bits on the output of the ADC.

ASIC Application-specific integrated circuit: An IC that is designed for a specific applications versus something like a microprocessor.

BBIC Baseband integrated circuit: An ASIC that has been designed specifically to process baseband information. BBICs can be considered the "Brains" in chipsets and often contain embedded microcontrollers, and the digital signal processing for communications.

CMOS Complementary metal oxide semiconductor: A semiconductor technology widely used in electronic components. CMOS typically uses less power than its predecessors.

CPRI Common pubic radio interface: The Common Public Radio Interface (CPRI™) is an industry cooperation aimed at defining a publicly available specification for the key internal interface of radio base stations between the Radio Equipment Control (REC) and the Radio Equipment (RE).

CtrlData Control data: Data that contains logistical information for command and control.

CtrlEn Control enable: A signal that is used to signify the start and end of the entire telegram in write operations and the address portion in read operations.

CtrlClock Control clock: The clock that is used for timing of CtrlData.

DUT Device under test

EDGE Enhanced data GSM environment: A technology that gives GSMA and TDMA similar capacity to handle services for the third generation of mobile telephony

8PSK 8 phase shift keying: A particular modulation format that has 8 separate phase states. The EDGE cellular system uses 8PSK.

EGPRS Enhanced general packet radio service: Enhanced data rates for the GSM cellular system.

EVM Error vector magnitude: A modulation quality metric widely used in communications. EVM is the magnitude of the error vector which points from where a symbol should lie in the constellation to where the symbol actually does lie.

FFT Fast Fourier transform: An efficient mathematical algorithm whose purpose is to transform sampled time waveforms into spectra.

Glossary

FPGA Field-programmable gate array: A programmable integrated circuit. Used often in initial designs for ASICs, as it can be reprogrammed to correct errors.

GMSK Gaussian minimum shift-keying: A form of frequency shift keying used in GSM cellular systems. Frequency shift keying uses two binary states to represent an analog waveform at a specific frequencies (a binary one signifies one frequency and a binary 0 a different frequency). The "G" refers to the fact that it uses Gaussian filtering. In GMSK, the tone frequencies are separated by one half the bit rate.

GPRS General packet radio service: A high-speed data service used by the GSM cellular systems. GPRS enables high-speed wireless Internet and other data communications (text messaging for example) over a GSM network.

GSM Global system for mobile communication: A standard for digital mobile telephony, based on a time-division multiple access frequency plan.

IC Integrated circuit: A chip, or die, which has a collection of transistors and electrical circuits built upon it. The name originated from the integration of previously separate components.

IQ In-phase and quadrature: In a modulated signal, I is considered the "real," or in-phase component, and Q the "imaginary" or "quadrature" component – the signal component that has been offset from the I signal component by 90 degrees (thus the term quadrature).

Modulation is measured by shifting the carrier frequency *fc* down to "0 Hz."

Given a signal $V(t) = A(t) * \text{Cos}\,[2\pi(fc)t + \theta(t)]$,

Where:
$V(t)$ is some modulated signal as a function of time
$A(t)$ describes the signal amplitude as a function of time
$2\pi\,(fc)t$ describes the carrier frequency as a function of time
$\theta(t)$ describes the phase as a function of time

We shift the carrier to zero hertz:

$$V(t) = A(t) * \text{Cos}\,[2\pi(\overset{1}{fc} - \overset{0}{fc})t + \theta(t)],$$

$= V(t) = A(t) * \angle\,\theta(t)$ (in phasor nomenclature)

$= V(t) = I(t) + jQ(t);$

and the signal is simply represented in standard nomenclature by using the terms "IQ."

MAC Media access control: In a computer OSI (Open Systems Interconnection) Model, there are seven layers. The MAC layer is a data communication protocol and is part of the data link layer. It is the sub-layer of the OSI model that provides addressing and channel access control for computer networking purposes.

Glossary

NZIF Near-zero intermediate frequency: A circuit where the IQ signals have been mixed up from DC to a low intermediate frequency.

OBSAI Open Base Station Architecture Initiative: A cellular base station interface specification to enable production of base station modules for interoperability between various module vendors.

PHY Physical layer: The first layer in the computer OSI model. It provides the transmission of bits through the network on an electrical and mechanical level.

RFIC Radio frequency integrated circuit: An ASIC that has been specifically designed to handle the radio frequency (RF) tasks of a chipset such as upsampling, mixing and certain types of filtering.

RxTxData Receive and transmit data: Nomenclature used in the DigRF 1.12 specification for IQ data.

RxTxEn Receive and transmit enable: A signal used by both RFIC and BBIC to indicate RxTxData transmission is to follow.

SDR Software defined radio: A radio communications system that uses programmable hardware which is controlled by software. The primary purpose of such a system is to enable switching between a variety of communications protocols.

SMA Subminiature A coaxial connector: A small coax cable connector that uses a threaded plug and socket (a screw-on type). The SMA is used in coaxial applications up to 18 GHz in frequency.

SMB Subminiature B coaxial connector: A very small snap-on coax cable connector, often used on circuit boards where test points are needed. The SMB is used in coaxial applications up to 4 GHz.

SysClk System clock: The system clock generated by the RFIC (in DigRF 1.12) to clock the RxTxData across the BBIC/RFIC interface.

SysClkEn System clock enable: An output from the BBIC to facilitate and power the SysClk.

TDMA Time division multiple access: A digital wireless transmission that allows multiple users to share a single frequency by assigning unique time slots. This prevents interference among users of the same frequency.

TTL Transistor-transistor logic: A type of logic circuit which uses bi-polar transistors to represent binary 0 or 1 which conform to specific voltage levels described by the TTL standard.

ViH Voltage input high: The designated high input voltage whose voltage level describes the threshold for a logic level 1.

ViL Voltage input low: The designated low input voltage whose voltage level describes the threshold for a logic level 0.

Glossary

VSA Vector signal analyzer: A signal analyzer that uses FFT processing versus swept-tuned analysis; essentially a digital spectrum analyzer.

XML Extensible markup language: A text format, based on SGML and designed by the World Wide Web Consortium to allow you to create a document for say, a web page, that can be easily understood by both computers and people. E.g. XML can translate into HTML, PDF, etc. giving you only one master document to edit.

ZeroIF Zero intermediate frequency: Also called "direct conversion," ZeroIF" refers to a radio technology that uses IQ signals instead of the older analog modulation technology, resulting in an initial intermediate frequency at 0 Hz.

For more information on the DigRF v3 Measurement Tools for Digital Serial Interface visit website **www.agilent.com/find/digrf**

Related literature

Publication title	Publication type	Publication number
Agilent 16900 Series Logic Analysis System Mainframes	Data Sheet	5989-0421EN
Agilent 89600 Series Vector Signal Analysis Software 89601A/89601AN/89601N12	Data Sheet	5989-1786EN
Agilent Logic Analyzers and 89601A Vector Signal Analysis Software	Technical Overview	5989-3359EN
How to Measure Digital Baseband and IF Signals Using Agilent Logic Analyzers with 89600 Vector Signal Analysis Software	Application Note	5989-2384EN

For copies of this literature, contact your Agilent representative or visit **www.agilent.com/find/dvsa**

Agilent Email Updates

www.agilent.com/find/emailupdates
Get the latest information on the products and applications you select.

Agilent Direct

www.agilent.com/find/agilentdirect
Quickly choose and use your test equipment solutions with confidence.

Agilent Open

www.agilent.com/find/open
Agilent Open simplifies the process of connecting and programming test systems to help engineers design, validate and manufacture electronic products. Agilent offers open connectivity for a broad range of system-ready instruments, open industry software, PC-standard I/O and global support, which are combined to more easily integrate test system development.

LXI

www.lxistandard.org
LXI is the LAN-based successor to GPIB, providing faster, more efficient connectivity. Agilent is a founding member of the LXI consortium.

Remove all doubt

Our repair and calibration services will get your equipment back to you, performing like new, when promised. You will get full value out of your Agilent equipment throughout its lifetime. Your equipment will be serviced by Agilent-trained technicians using the latest factory calibration procedures, automated repair diagnostics and genuine parts. You will always have the utmost confidence in your measurements.

Agilent offers a wide range of additional expert test and measurement services for your equipment, including initial start-up assistance onsite education and training, as well as design, system integration, and project management.

For more information on repair and calibration services, go to

www.agilent.com/find/removealldoubt

www.agilent.com

For more information on Agilent Technologies' products, applications or services, please contact your local Agilent office. The complete list is available at:

www.agilent.com/find/contactus

Phone

Americas	
Canada	(877) 894-4414
Latin America	305 269 7500
United States	(800) 829-4444

Asia Pacific	
Australia	1 800 629 485
China	800 810 0189
Hong Kong	800 938 693
India	1 800 112 929
Japan	81 426 56 7832
Korea	080 769 0800
Malaysia	1 800 888 848
Singapore	1 800 375 8100
Taiwan	0800 047 866
Thailand	1 800 226 008

Europe	
Austria	0820 87 44 11
Belgium	32 (0) 2 404 93 40
Denmark	45 70 13 15 15
Finland	358 (0) 10 855 2100
France	0825 010 700
Germany	01805 24 6333*
	*0.14€/minute
Ireland	1890 924 204
Italy	39 02 92 60 8484
Netherlands	31 (0) 20 547 2111
Spain	34 (91) 631 3300
Sweden	0200-88 22 55
Switzerland	(French)
	44 (21) 8113811 (Opt 2)
Switzerland	(German)
	0800 80 53 53 (Opt 1)
United Kingdom	44 (0) 7004 666666

Other European countries:
www.agilent.com/find/contactus

Revised: March 23, 2007

Product specifications and descriptions in this document subject to change without notice.

Printed in USA, May 8, 2007
5989-5290EN

Agilent Technologies