New Features in V93000 EDGE Demodulation

Joe Kelly and Max Seminario, Verigy

Abstract

This work presents an overview of some of the new features pertaining to EDGE found in the V93000 Demodulation Library. These can improve test times and TTM (Time To Market) by allowing users to select demodulation results for either a specific, or multiple EDGE frames (sometimes, loosely referred to as bursts). Code examples are also provided. It is assumed that the reader is somewhat familiar with the basics of coding demodulation procedures in the V93000 programming environment and as such, details on the structure of that implementation are not provided. For more general demodulation implementation details on the V93000 it is suggested to view the V93000 documentation.

Demodulation on the V93000

The demodulation library within the V93000 supports all of the modulation formats found in consumer wireless devices and is essential to finding TX (transmit) and RX (receive) EVM values. Many consider demodulation algorithms to be complex. However, the process can be broken down into a few key high-level pieces to make it better understood.

A flow chart of the algorithm is shown in Figure 1. To use the demodulation library, first an array, either from a digitized capture or from a file, is acquired. This sampled I/Q array of information is applied to the algorithm. Each waveform has its own properties such as sample rate, modulation type, encoded data, etc. and from them the appropriate EVM can be calculated. The user selects the demodulation type and required input parameters and from there, the EVM or other needed values are calculated.

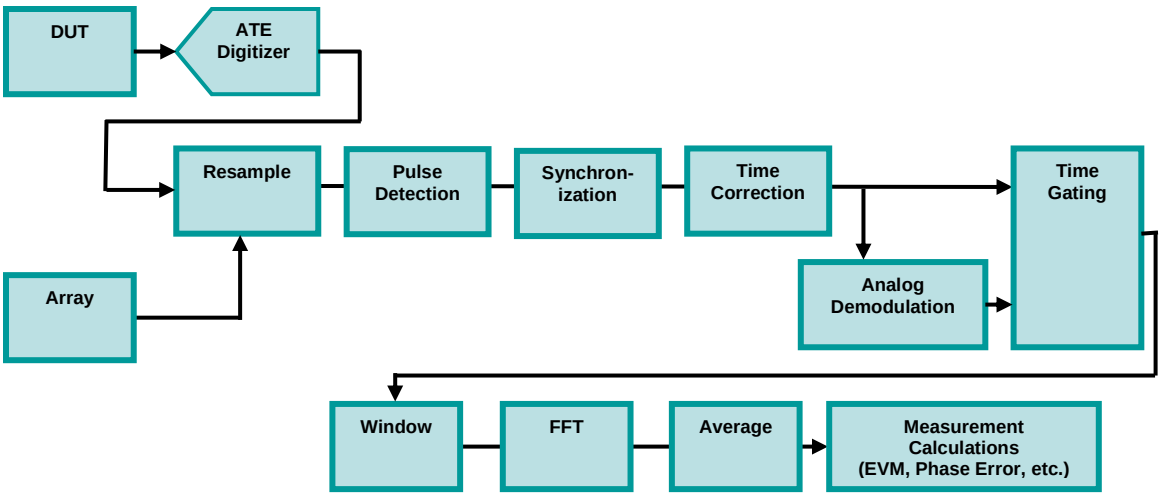

Figure 1 Flow chart of the algorithm used for the demodulation library on the V93000 [1].

The following outline of the key steps within the algorithm reference Figure 1 and how it handles an array of data (on the V93000, data is passed into the algorithm as an array of doubles, either ARRAY_D or ARRAY_COMPLEX). This is usually a capture of the signal coming from the DUT (Device Under Test) by the digitizer (DGT), but it can be achieved by reading digital bits as well and converting to equivalent analog levels. Alternatively, it can be a mathematically generated waveform or saved capture from a separate piece of hardware/ATE. Supported file formats are ASCII, Agilent Signal Studio encrypted, and Agilent 89600 recordings. The steps of the algorithm are:

- Resample - Data is internally resampled to a sampling rate where corrections can be made accurately and test times are minimized. The value is a function of each standard's symbol rate or chip rate for the code domain multiple access standard types.

- Pulse Detection - Many standards, to save test time, will detect whether a pulse is present or not prior to synchronization. Synchronization is a time consuming process when searching for the sync (synchronization) word and it is a lot less time intensive to first search if there is a valid pulse to begin with. With pulse detection a threshold level is set. This level defines what an on/off pulse is. To detect the burst there has to be an OFF/ON/OFF period. If a signal contains no OFF period pulse it is suggested to have the pulse detection set to off and only search for a synchronization word.

- Synchronization - Synchronization words have a known bit sequence that is defined by the standard. Synchronization words are made to be robust to avoid symbol errors and many standards will have this synchronization word be at the constellation where the least symbol errors will occur. An internally generated synchronization word is created and the signal is searched until a match with the synchronization word is found (or a best-fit correlation occurs). Once the sync word and the digitized signal are aligned and we know exactly how many samples in this peak happens at, we can overlay both signals and we will know exactly

where we are in the received signal. The most commonly used input parameter in the demodulation library that is associated with controlling synchronization is syncMode.

- Time Correction – Corrections are applied. For example, once synchronization occurs the demodulator knows exactly how off the received signal is in phase, magnitude, equalization, etc. (phase due to transmission line length and local oscillator phase, magnitude due to gains or losses in the path and timing due to the start time of sampler and transmission line length).

- Analog Demodulation (Optional) – This provides AM, FM, and PM analog demodulation for analysis of either intentional or unintentional modulation contained within a signal.

- Time Gating – Time Gating is implemented when the user is interested in a specific part of the received signal or packet. For example, time gating would be if the signal received is composed of several different types of waveforms (maybe first half is a 1KHz sine wave and second half is a 2KHz sine wave). If the user wants to analyze the first half of this combined waveform only they would use time gating to ensure that only the 1KHz part of the received waveform is analyzed. For waveforms that are standard compliant, this is used to look at certain parts of the received packet (automatically done in standard compliant waveforms for measuring EVM).

- Window - Windowing is performed prior to obtaining an FFT (Fast Fourier Transform) on a signal. The true frequency domain response of a certain signal is obtained if the signal is infinitely long. Any truncation of this signal in time will convolve the frequency response of a rectangular window that is multiplying the signal. This truncation is calculated by the RBW (resolution bandwidth) set in the frequency domain measurement. A rectangular window (or no windowing) has a lot of power in its sidelobes so windowing with a different shape helps obtain the best frequency response representation of the true signal. The number of samples used for the FFT for a certain RBW depends on the ENB (Equivalent Noise Bandwidth) of the specific window being used. This is, RBW = ENB*Fs/N, where Fs is the sampling frequency, N is the number of samples and ENB is the Equivalent Noise Bandwidth.

- FFT – Fast Fourier Transform. This is implemented to look at the energy content of a signal in the frequency domain. For demodulation FFTs are performed to equalize the signal or to extract the received symbols (such as in an OFDM type of signal).

- Average – Averaging is a way to compute the mean of a measurement (mean = $\Sigma x(n)/N$). Averaging of FFTs is a great way to lower white noise since white noise has a phase that is randomly uniformly distributed. Due to this property, if

enough averages are implemented, the white noise will sum up to a very small amount of power due to the random nature of the phase.

- Measurement Calculations – – Mathematical derivations on a signal in order to extract certain signal properties (or parameters). This could be something as simple as signal power or as complex as EVM.

EDGE Demodulation

As with the Agilent 89600 Vector Signal Analyzer (VSA) software, the V93000 offers two primary formats/classes for EDGE demodulation. They are:

- EDGE_FLEX – Demodulates 2G 8PSK EDGE signals
- GSM_EDGE_EVO – Demodulates 2G 8PSK EDGE signals as well as EDGE Evolution signals

A new feature in both of these classes is the ability to perform multi-frame demodulation. From a production test standpoint, this saves both, hardware memory as well as test time. Capture memory in the hardware is saved because with one-frame-at-a-time demodulation, leading samples are required to ensure that the frame start point is found. Test time is saved because synchronization only has to be performed for the first of any number of frames, compared to multiple single frame analyses where each frame would have to be synchronized by the algorithm.

Both of these classes have their time and place. Just as in the Agilent 89600 VSA, EDGE_FLEX is based on a generic digital demodulation algorithm and as such, provides a great amount of flexibility. GSM_EDGE_EVO was designed around and built upon solely the GSM and EDGE standards and also incorporates the ability to demodulate the newer EDGE Evolution higher data rate standard.

Multi-Frame Demodulation with EDGE_FLEX

As always, EDGE_FLEX can simply demodulate a single frame and provide a myriad of results including EVM, frequency error, phase error, and even the demodulated bit sequences. If it is desired to demodulate more than one frame, this is performed in EDGE_FLEX by loading the waveform to be analyzed and then looping the demodulation through the waveform. Upon each iteration of the loop sequence, a new starting point of analysis is defined. This is shown in the code below where 20 frames are demodulated in a loop. With each analysis, the endpoint (*burstEndIndex*) becomes the starting point of the subsequent frame's analysis. With each iteration of the loop, the EVM is obtained (*symbolRmsEVM*) and its value logged and tested or stored to provide a user-calculated average and worst-case EVM over all of the frames.

```
double evmRms = -999.9;  // Initialize to a ridiculous value
double sampleRate;       // User must provide a value for this
ARRAY_COMPLEX iqData;    // User must populate this array

int burstEndIndex = 0; // Defines start point for subsequent burst searches

static DEMODULATION demod("EDGE_FLEX");

for (i=1; i<=20; i++)
{
  demod.setInputParameter("inputDataOffset", burstEndIndex);

  if (i==1) demod.execute(iqData, sampleRate);
  else demod.execute(); // Skips synchronization to save test time

  demod.getResult("symbolRmsEVM", evmRms);
  demod.getResult("burstEndIndex", burstEndIndex);

  cerr << "EVM (rms, %), Frame " << i << ": " << evmRms << endl;
}
```

Multi-Frame Demodulation with GSM_EDGE_EVO

When demodulating multi-frames with GSM_EDGE_EVO, the algorithm is more integrated compared to that of EDGE_FLEX. No user-level looping through the frames is required. The following code shows this. Rather, the user provides the waveform array, length of a frame (*frameTime*), and number of frames to be analyzed (*numberOfFrames*).

After the execution, results such as EVM, frequency error, phase error, and many others are available. Referring to the code below, these are available typically in three formats, average of all frames analyzed (e.g., *symbolRmsEVM*), worst-case over all frames analyzed (e.g., *symbolRmsEVMWC*), and as a vector of length *numberOfFrames* containing the value for each frame analyzed (e.g., *symbolRmsEVMV*). An example of how to access/print the vector results is provided in the code.

```
double evmRms = -999.9;  // Initialize to a ridiculous value
double sampleRate;       // User must provide a value for this
ARRAY_COMPLEX iqData;    // User must populate this array

static DEMODULATION demod("GSM_EDGE_EVO");

demod.setInputParameter("modulationSchemeAutoSelect", FALSE);
demod.setInputParameter("modulationSchemeManual", 1); // 1=8PSK
demod.setInputParameter("frameTime", 0.001329); // Frame length in seconds
demod.setInputParameter("numberOfFrames", 20);

demod.execute(iqData, sampleRate);

demod.getResult("symbolRmsEVM", evmRms);       // EVM (rms, %) over all frames
demod.getResult("symbolRmsEVMWC", evmRmsWC);   // EVM (rms, %, worst-case)
demod.getResult("symbolRmsEVMV", evmPerFrame); // EVM (rms, %) per frame
```

```
cerr << "EVM (rms, %), averaged over 20 frames: " << evmRms << endl;
cerr << "EVM (rms, %), Worst case: " << evmRmsWC << endl;

//Print out per-frame results
for (i=0; i<20; i++)
{
  cerr << "EVM (rms, %), Frame " << i+1 << ": " << evmPerFrame[i] << endl;
}
```

Graphing Results within the Demodulation Library

The demodulation library has graphs available for each demodulation class. These can be used for debugging as well as within the Signal Analyzer Tool. The generateGraph() function can be used with any of the many available graphs listed in Table 1. Using an IQ constellation plot as an example, a graph can be displayed in the Signal Analyzer Tool via the following code.

```
double sampleRate;  // User must provide a value for this
ARRAY_COMPLEX iqData;       // User must populate this array
DEMODULATION demod("GSM_EDGE_EVO);
demod.execute(iqData, sampleRate);
demod.generateGraph("TestSuite", 1, 1, "PinName", "Label", TM::CONSTELLATION);
```

More details on the generateGraph() function can be found in the V93000 documentation.

The GSM_EDGE_EVO class has additional available graphs for some of the other functionality such as Output RF Spectrum (ORFS) analysis. Those are outside the scope of this article, but will be covered in an upcoming article.

Parameter	Description
COMPOSITE	Displays the composite EVM results, this often has a very irregular shape to the plot
CONSTELLATION	Displays the I/Q constellation
EVM_VS_SYMBOL	Displays the rms EVM value on a per-symbol basis
POWER_VS_TIME	Displays the uncalibrated power as a function of time, useful for observing the shape of the waveform in the time domain, also displays an overlay and values showing the beginning and end of the analysis region
SPECTRUM	Displays the uncalibrated power as a function of frequency

Table 1 Available graphs in EDGE_FLEX and GSM_EDGE_EVO classes.

Summary

A description of the algorithms used within the V93000 Demodulation Library is presented. A few key use models when using the demodulation classes EDGE_FLEX and GSM_EDGE_EVO, were shown along with the associated test method code so that the reader has a place to begin writing their code for demodulating EDGE signals.

References

[1] Agilent Technologies, “Agilent 89600 Vector Signal Analysis Software,” Technical Overview 5989-1679EN (2010).