

AN ITU-T G.994.1 PROTOCOL ANALYSIS TOOL FOR ADSL

Matthew J. Langlois[†], Michael J. Carter^{*}, Scott A. Valcourt[†], William Lenhearth^{†*}

[†] InterOperability Laboratory, 121 Technology Drive, Suite 2, University of New Hampshire, Durham, NH 03824, USA.

Email: mjl@iol.unh.edu, sav@unh.edu, whl@iol.unh.edu

^{*} Department of Electrical and Computer Engineering, University of New Hampshire, Durham, NH 03824, USA.

Email: mike.carter@unh.edu, whl@iol.unh.edu

ABSTRACT

One of the most commonly deployed DSL variants is ADSL. Despite this fact, deployment of new robust and reliable ADSL services is increasingly difficult due, in part, to the physical limitations of the copper telephone system infrastructure, and also in part to the lack of useful ADSL network debugging tools. ADSL service providers and technicians currently lack a device capable of decoding physical layer signaling and displaying actual physical layer parameters and statistics associated with a live ADSL connection, independent of the end stations. Similar devices used in other network technologies are often referred to as protocol analyzers. The intent of this paper is to illustrate how Matlab [1], in conjunction with a DSP or a PC, can be used to create an effective ADSL handshaking protocol analyzer based on ITU-T G.994.1 (G.hs) [2]. G.hs conformance is critical in establishing a successful ITU-T G.992.1 (G.dmt) [3] based ADSL connection.

KEY WORDS

DSL, Broadband, Protocol Analysis

1. INTRODUCTION

The demand for high-speed data networks in the “last mile” has driven the need for robust, interoperable, and easy to use multi-vendor Digital Subscriber Line (DSL) access solutions. DSL technology is attractive because it requires little to no upgrading of the existing copper infrastructure that connects nearly all populated locations in the world. There are many variations of DSL, each aimed at particular markets, all designed to accomplish the same basic goals. ADSL, or Asymmetric DSL, is aimed at the residential consumer market. ADSL provides higher data rates in the downstream direction, from the central office to the end user, than in the upstream direction, from the end user to the central office. Within the Internet connectivity-based residential environment, small requests by the end user often result in large transfers of data in the downstream direction.

ADSL is a direct result of the asymmetric nature of the Internet and the needs of the end user, and was originally designed for video-on-demand applications.

ADSL is point-to-point technology. Prior to an active connection between the central office (CO) ATU-C (ADSL transceiver unit – central) and customer premise equipment (CPE) ATU-R (ADSL transceiver unit – remote), the two devices must determine the characteristics of the loop that connects them (the local loop). This process is known as initialization and allows ADSL systems to adapt the conditions of the local loop. Initialization consists of four parts: handshaking, transceiver training, channel analysis, and exchange. ITU-T G.992.1 (G.dmt) defines discrete multi-tone (DMT) based ADSL, including all aspects of initialization except for handshaking. Handshaking is formally defined in its own specification, ITU-T G.994.1 (G.hs). G.hs marks the beginning of initialization, and defines how two ADSL devices acknowledge each other, exchange capabilities, and determine values for physical layer parameters.

2. ARCHITECTURE

In terms of ADSL, a physical layer protocol analyzer should have two major components: a non-intrusive line tap, which allows the protocol analyzer to be transparent to the medium and any other devices using that medium, and the hardware and software required to demodulate/decode the physical layer signaling. Figure 1 shows a block diagram of one possible configuration for a DSL protocol analyzer.

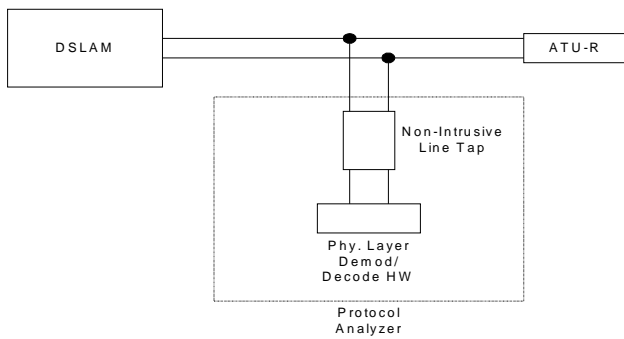


Figure 1. Protocol analyzer block diagram.

Such an analyzer can be used as a stand-alone interoperability debugging tool, in conjunction with an ADSL emulator as a conformance verification tool, or simply as a line-monitoring tool. It should be noted that this tool is not limited to G.dmt based ADSL, since G.hs is used in many different ITU DSL physical layer specifications. As a result, G.hs sessions, regardless of technology, can be debugged and verified using such an implementation.

As a basic interoperability debugging tool, this G.hs analyzer can be used to effectively identify and report any problems that may arise during an incomplete or unsuccessful G.hs session between two devices. In addition, all G.hs signals, timing constraints, and state transitions can be analyzed and verified. By default, this functionality identifies interoperability problem areas. As a conformance tool, this analyzer can be used with an ADSL or SHDSL [4] emulator, or similar device capable of generating G.hs signals, to completely stress and test the G.hs functionality of another ADSL or SHDSL device (this conformance type functionality can be extended to cover all of initialization as well). As a conformance tool, this analyzer provides insight into problem areas before they are actually discovered in the field.

Regardless of implementation, there are two fundamental hurdles that must be cleared – an ADSL physical layer protocol analyzer must have some means by which it: 1) Receives and/or captures waveform data, and 2) Demodulates and translates the received data. Receiving data is related to the signal extraction architecture that is utilized, of which there are two basic types: in-line and tapped. An in-line architecture is one in which the ADSL signals of interest physically pass through the analyzer. In this type of setup, the analyzer becomes part of the circuit and essentially splits a single loop into two distinct loops. In contrast, a tapped architecture is one in which the ADSL signals of interest do not pass through the analyzer but are instead received and/or captured using a non-intrusive tap or probe. The latter is very similar to the traditional method of voltage measurement using a scope and voltage probe. Figure 2 shows the two types of tapping architectures.

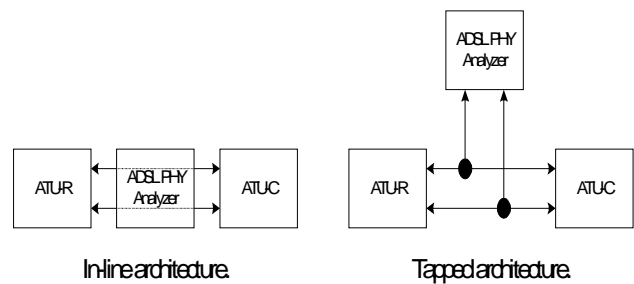


Figure 2. Architecture types.

The goal of either architecture is to essentially pass all ADSL signals without altering or changing the characteristics of the line and/or the actual ADSL signals themselves, while providing a connection point or interface for demodulation and translation functions. This requirement is easier to implement with a tapped architecture because, as mentioned previously, the ADSL signals of interest do not physically pass through the device. A properly designed tapped architecture also inherently has little to no effect on the characteristics of the line and affords the use of proven existing equipment designed for these purposes, whereas an implementation based on an in-line architecture would have to be designed, constructed, and tested from the ground up.

3. IMPLEMENTATION

One way to implement a tapped architecture is through the use of an active differential probe and a digital storage oscilloscope (DSO). This combination provides a stable means by which an ADSL line can be tapped and the signals of interest stored for post-processing and analysis. Another option is to replace the DSO with dedicated hardware that demodulates and decodes the data on the line in real-time. The tradeoff between the two approaches is complexity versus speed. In the prototype stage however, analysis of the physical layer signals is most easily accomplished in software, using a programming environment like Matlab [3]. A software-based approach provides more flexibility and debugging capabilities than the hardware-based approach. The logical “next step” for most proven software-based designs is to develop a hardware implementation. For these reasons the implementation outlined in this chapter is based on a tapped architecture using an active differential probe, DSO, and software-based post-processing analysis. In this case, the analysis software, based on a three-part physical layer analysis architecture shown and implemented in Matlab, has a fairly quick execution time, and is very adaptable in the sense that it provides a solid basis to which extended functionality and capability can be added at a later point.

A practical physical layer analysis architecture has three basic parts: demodulation, translation, and analysis. Demodulation is required to determine the transmitted binary data. Translation is required to parse

and convert the binary output of the demodulation block into G.hs signals and messages. Once the signals and messages have been identified, state labels can be assigned. As part of the translation process, timing information must also be maintained. Analysis is required to extract useful information from the translated state labels and to verify device conformance with the guidelines specified in G.hs. Useful information comes in the form of physical layer device parameters, capabilities, and supportable modes of operation that are transmitted during the message transaction portion of G.hs. Device conformance requires tracing through each state diagram and verifying state transitions and timing constraints. Figure 3 shows a basic physical layer protocol analyzer architecture.

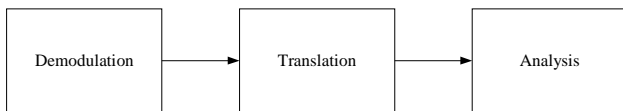


Figure 3. Physical layer analyzer architecture.

3.1. Setup and capture

The A43 carrier set will be the focus of this discussion because it is the most common. However, the same methodology can be applied to all carrier sets. Using this test setup, a LeCroy WavePRO 950 1GHz DSO, with 32 Mpoint capture memory, is used to capture and store a complete G.hs session. An active differential probe, such as the LeCroy AP 034 or the HP/Agilent 1141A, is required to interface the DSO to the physical copper wires connecting the HSTU-x's and carrying the G.hs session. Using a program distributed by LeCroy called ScopeExplorer, data can be transferred from the DSO across a LAN to a PC (with Matlab and ScopeExplorer installed). The PC actually performs the G.hs demodulation using a Matlab script developed for this project. Figure 4 shows the test setup required for G.hs physical layer analysis using the implementation outlined in this paper.

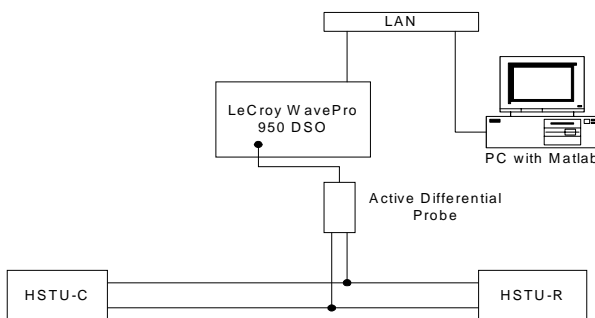


Figure 4. Physical layer protocol analyzer test setup.

Based on the highest carrier frequency of the A43 carrier set, $64 * 4312.5\text{Hz} = 276 \text{ kHz}$, the sampling

rate of the DSO must be set to at least 552 kHz (1 Msample) per second (or higher, although 1 Msample per second, which is comfortably above the Nyquist rate, is used throughout this discussion). A complete G.hs session is approximately 2 to 6 seconds in duration. At 1 Msample per second, the DSO should be configured to capture at least 10 million points, which is 10 seconds worth of data. At a resolution of 8 bits, the captured waveform results in a file size of about 10 Mbytes. The DSO can be configured to trigger on a certain voltage level, indicating that a G.hs session is taking place, or the DSO can be triggered manually just before the G.hs session takes place. Along with a G.hs session, at least 4 seconds of other data will be captured and stored in the DSO. This remaining data is usually other parts of the initialization sequence and can, for the purposes of G.hs analysis, be ignored. However, it is important to realize which part of the captured data is actually the G.hs session and which part is not.

The original Matlab scripts used to transfer, display, and demodulate the captured G.hs sequences were written using Matlab 5.3.1 and Simulink 3, with the Signal Processing and Communication Toolboxes, as well as the DSP and Communication Blocksets for Simulink. The current software version, v3.1 as seen in Figure 5-3, has been updated for use with Matlab 6.1 and Simulink 4, and includes a custom graphical user interface (GUI) for control of the demodulation and presentation of the recovered data. The majority of the G.hs demodulation is done in Simulink, and due to the high sampling rates used and the corresponding large waveform files, CPU speed plays a large role in the overall execution time (execution time refers to the amount of time required for this G.hs analysis tool to produce an output).

3.2. Demodulation

The first step in the demodulation process is to decouple the individual carrier frequencies so that they can be demodulated independently. Six independent bandpass filters, one for each carrier frequency in the A43 carrier set, have been designed for this purpose in Simulink. Each of these six BPF's is a fourth order Butterworth IIR filter with a 10kHz bandwidth. This particular filter was chosen for its stability and flatness over the pass band. The 10kHz bandwidth was chosen to satisfy the requirement stated in Section 6.3.1 of G.hs. Section 6.3.1 states that: "For the 4.3125 kHz signalling family, the transmit filter shall have a bandwidth such that all of the -3 dB points of the filter shall have frequencies which differ at least 4.3125 kHz from any G.994.1 carrier frequency used." Simulink's "Digital IIR Filter Design" block, available with the DSP Blockset, was used to generate each of the six individual BPF's within the decoupling filter bank. Frequency normalization is accomplished by dividing the frequency of interest by the effective Nyquist frequency, which is half the sampling rate (1 Msample/second), or 500 kHz.

The next step in the G.hs demodulation process is to individually demodulate the separate modulated carrier frequencies. Recall that the modulation scheme used throughout G.hs is DBPSK, therefore six separate DBPSK demodulator functions will be required. The decoupled output of each of the individual BPF's will be fed into individual DBPSK demodulator functions. There are essentially six separate demodulation systems running in parallel. Simulink has a built-in DBPSK demodulator block included as part of the Communications Blockset, termed "M-DPSK Demodulator Passband." This block will be used as the primary demodulation engine. Relevant arguments of the "M-DPSK Demodulator Passband" block include: M-ary number (2 in this case), carrier frequency, symbol period, sample time, and baseband samples per symbol. These arguments are defined by the carrier frequency and signaling family of interest. For example, the lowest carrier frequency within the A43 carrier set is $9 * 4312.5\text{Hz} = 38812.5\text{Hz}$. The symbol rate for the A43 carrier set (which is part of the 4.3125kHz signaling family) is $4312.5 / 8 = 539.0625$ symbols per second. The sample time is the sample rate of the DSO, which in most cases is 1 Msample/second. Thus, for each of the six DBPSK demodulator functions the only arguments that will differ are the carrier frequency and the "Baseband samples per symbol."

The "Baseband samples per symbol" argument refers to the frequency downconversion factor used by the "M-DPSK Demodulator Passband" block. The "M-DPSK Demodulator Passband" block simply performs downconversion on the incoming signal and passes it to the Simulink "M-DPSK Demodulator Baseband" block. Downconversion is also known as bandpass sampling, and is similar to converting from an RF signal to an IF signal, albeit at zero IF, using a mixer in analog communications systems. Downconversion throws samples away, effectively "re-sampling" the signal at a lower sampling rate. For this particular case, the advantage to downsampling (which is not required) is that the effective reduced sampling rate and decreased number of samples dramatically improves the execution time.

Simulink's "M-DPSK Demodulator Passband" block consists of two basic parts: frequency downconversion, and the "M-DPSK Demodulator Baseband" block, which actually performs the bulk of the demodulation. Version 4 of Simulink does not allow access to the sub-blocks of the "M-DPSK Demodulator Baseband" block using the "look under mask" command. However, Version 3 of Simulink does allow access to the "DPSK Passband Demodulator" and related blocks. Assuming that the blocks in Version 4 of Simulink are based on the blocks in Version 3, Matlab/Simulink utilizes the correlation method of DPSK detection.

The correlation method of detection essentially compares the received modulated signal (and noise) to all of the possible noiseless outcomes and chooses the closest

match. This is performed as such: the received signal is multiplied with an array of sinusoids, each differing in phase according to the possible phase outcomes (i.e. for $M = 4$, there would be four sinusoids, each differing in phase by 90°), at the carrier frequency of interest. Each resulting product is fed into an integrator that resets to zero at the beginning of every symbol period. The output of the integrator, which is the correlation of the received noisy signal and all of the possible modulated outcomes, is then fed into a mapper, which maps correlation levels to binary outputs. Since a DBPSK modulator has only two possible outcomes, the DPSK demodulator has only one correlation function – if the correlation of the two input signals is large enough a binary 0 or 1 is the decision, and if the correlation is not large enough, the opposite binary digit is the decision. Recall that exact phase synchronization with the carrier frequency is not required with DPSK receivers/demodulators because the data is encoded as relative phase differences from one symbol to the next. Therefore, the final stage in the demodulation process is a differential decoder that extracts the message bits from the demodulated differentially encoded binary sequence.

Each of the six individual DBPSK demodulators produces a binary data stream. Recall that all carrier frequencies within a particular upstream or downstream carrier set defined in G.hs (i.e. all upstream and all downstream carrier frequencies) are modulated with the same data. With three carrier frequencies, a median filter can be used to determine the 'majority' result of the individual binary output streams because, for an odd number of inputs, Simulink's "Median" block sorts all of the inputs by value and outputs the middle row. This function, as implemented by Simulink, is known as majority logic decoding. Thus, median filtering in Simulink produces a single bit stream that is capable of eliminating single bit errors. Figure 5a shows the Simulink model required to demodulate one of the six carrier frequencies defined in the A43 carrier set. The complete Simulink model required to demodulate all carrier frequencies within the A43 carrier set uses six identical demodulation blocks each with a different bandpass filters tuned to the frequency of that particular carrier. The output of the first three demodulation blocks are fed into a median filter, shown in Figure 5b, which produces the upstream direction majority output. Likewise, the output of the last three demodulation blocks are fed into a median filter, similar to that shown in Figure 5b, which produces the downstream direction majority output.

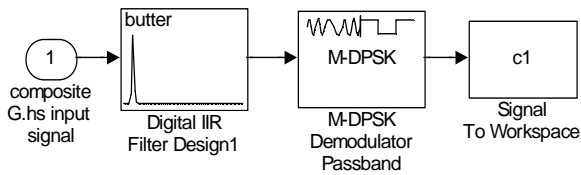


Figure 5a. Simulink G.hs demodulation model for a single carrier frequency.

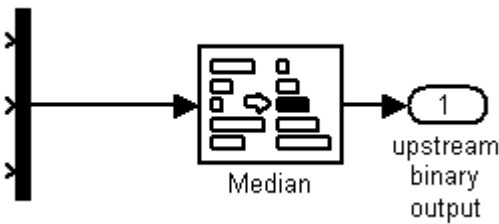


Figure 5b. Simulink G.hs demodulation model for the upstream median filter operation.

3.3. Translation

Once started, the six DBPSK demodulator functions run continuously for a fixed length of time determined by the user, i.e. the demodulators do not stop when DBPSK data is not present. This tool is used to demodulate an entire G.hs session from start to finish. Thus, different binary patterns can be expected in the output during different states of G.hs. With this in mind, each of the five signals defined in G.hs (TONES-REQ, SILENT, TONES, FLAGS, and GALFS) have a distinct DBPSK demodulated binary output pattern.

TONES-REQ consists of transmitting on all carrier frequencies with phase reversals (180° phase shifts) every 16ms. With a symbol period of $8/4312.5 \cong 1.855\text{ms}$, eight or nine symbols (because $16 / 1.855 \sim 8.62$) will be demodulated between phase reversals. According to the differential encoding algorithm mentioned previously (recall that a binary 0 in the output represents no phase difference and a binary 1 represents 180° phase difference between adjacent outputs), this translates into a binary output sequence that looks like: [. . . 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 . . .].

The TONES signal consists of transmitting power on all carrier frequencies continuously without phase modulation, thereby in a binary output sequence of 0's. The SILENT signal is defined as zero output voltage from the HSTU-x's, resulting in a random binary output, because the DBPSK demodulators are essentially demodulating line noise. The FLAGS and GALFS signals are defined as continuously modulated HDLC FLAG (0 1 1 1 1 1 0) and GALF (1 0 0 0 0 0 1) octets, respectively. Also, the resultant bit patterns of all fields

and parameters contained within the message transaction portion are formally defined in G.hs and follow the bit-encoding algorithm defined in G.hs. Figure 6 shows how certain binary outputs can be translated to known G.hs states.

G.hs State	Expected Demodulated Binary Output													
FLAG (1 octet) (C or R)	0	1	1	1	1	1	1	0						
GALF (1 octet) (C or R)	1	0	0	0	0	0	0	1						
TONES (C or R)	0	0	0	0	...	0								
TONES_REQ (R only)	1	0	0	0	0	0	0	0	NOTE: 7 OR 8 zeros may follow the 1.					
SILENT (C or R)/Unknown	Random – binary output is demodulated noise.													

Figure 6. Translation table including possible known binary outputs and associated G.hs states.

3.4. Analysis

The G.hs demodulator described above is embedded within a larger Matlab script file that handles data input and output and basic device control functionality. The demodulated binary output produced by the Matlab based G.hs demodulator is written to a text file, a sample of which is shown in Figure 7, which can then be parsed for all of the state transition information and timing constraints of both devices.

Figure 7 is a sample of actual results obtained from two devices. The output text file consists of nine columns of data. The first three columns represent the binary output of each of the three carrier frequencies in the upstream portion of the A43 carrier set. The fourth column is the median filtered output of the first three columns. The fifth, sixth, and seventh columns are the binary output of each of the three carrier frequencies in the downstream portion of the A43 carriers set. The eighth column is the median filtered output of the fifth, sixth, and seventh columns. The last column is the time stamp and indicates the time, as measured from the beginning of the captured sequence, at which each of the binary outputs in that row occurred.

Notice that the first four columns in Figure 7 are the same, indicating that all three carrier frequencies in the upstream direction are modulating the same data. Using Figure 6, it can be shown that the upstream carrier frequencies indicated in Figure 7 are modulating the R_TONES_REQ signal, which in turn indicates that the HSTU-R is in the R_TONES_REQ state. For clarification purposes, a device in a particular state continuously transmits the same signals while in that state. For example, a device in the R_TONES_REQ state continuously modulates and transmits the R_TONES_REQ signal, as shown in Figure 6. Notice that the fifth, sixth, seventh, and eighth columns are not the

same, which indicates that the HSTU-C is either in the C-SILENT state or in unknown state.

This translation from binary output to known G.hs states and corresponding signals, as shown in Figure 6, is the first step in the analysis procedure. Following this basic translation, the three phases of G.hs can be identified: start-up, message transactions, and clear down. In addition, G.hs frame boundaries and all basic and extended message transactions can be identified, and all message fields and parameters can be translated according to the tables in G.hs.

Complete analysis of the translated binary output text file must be done by hand, but can be done quickly and efficiently providing in depth information about the state transitions of both the HSTU-R and the HSTU-C and about the message transactions between the two endpoints. This information can be used to verify timing constraints, device capabilities, and conformance to the G.hs specification. Detailed G.hs analysis identifies problem areas before they arise in real world equipment and real world situations, and reduces the likelihood of interoperability issues in a multi-vendor environment.

0	0	0	0	1	1	0	1	1	0.017130
0	0	0	0	0	0	0	0	1.018986	
0	0	0	0	1	1	1	1	1.020841	
1	1	1	1	1	1	1	1	1.022696	
0	0	0	0	0	0	1	0	1.024551	
0	0	0	0	1	1	1	1	1.026406	
0	0	0	0	1	1	0	1	1.028261	
0	0	0	0	1	1	1	1	1.030116	
0	0	0	0	0	1	1	1	1.031971	
0	0	0	0	1	1	0	1	1.033826	
0	0	0	0	0	0	0	0	1.035681	
0	0	0	0	1	0	0	0	1.037536	
1	1	1	1	0	0	0	0	1.039391	

Figure 7. Sample of the binary output text file created by the G.hs demodulator.

4. CONCLUSION

This G.hs analysis tool can be used to identify all interoperability issues associated with G.hs sessions between two devices. This functionality has been proven in real-world situations with real-world devices and networks in the University of New Hampshire Research Computing Center InterOperability Laboratory. In addition, the implementation outlined in this paper is adaptable and provides an architecture in which provisions for G.dmt Annex B, G.dmt Annex C, SHDSL, and VDSL compliant devices can be easily added. Work is currently underway in the InterOperability Laboratory to extend the capabilities of the G.hs protocol analysis

tool to cover these other technology areas, and to this point, provisions for testing of SHDSL G.hs sessions based on this architecture have been completed.

5. ACKNOWLEDGEMENTS

The work described here was performed in partial fulfillment of the requirements for the Master of Science in Electrical Engineering degree at the University of New Hampshire, Durham, NH USA.

REFERENCES

[1]The MathWorks. All Matlab Version 5.3.1 and 6.1 and Simulink Version 3.0 and 4.0 help files associated with the functions and blocks used in this document, including the help files for use with the DSP and Communications Toolboxes for Matlab and the DSP and Communications Blocksets for use with Simulink.

[2]International Telecommunication Union Standardization Sector (ITU-T). Series G: Transmission Systems and Media, Digital Systems and Networks: Digital section and digital line system – Access networks. ITU-T Pre-Published Recommendation G.994.1, Handshake procedures for Digital Subscriber Line (DSL) Transceivers. February 2001.

[3]International Telecommunication Union Standardization Sector (ITU-T). Series G: Transmission Systems and Media, Digital Systems and Networks: Digital section and digital line system – Access networks. ITU-T Recommendation G.992.1, Asymmetric digital subscriber line (ADSL) transceivers. June 1999.

[4]International Telecommunication Union Standardization Sector (ITU-T). Series G: Transmission Systems and Media, Digital Systems and Networks: Digital section and digital line system – Access networks. ITU-T Pre-Published Recommendation G.991.2, Single-Pair High-Speed Digital Subscriber Line (SHDSL) Transceivers. February 2001.