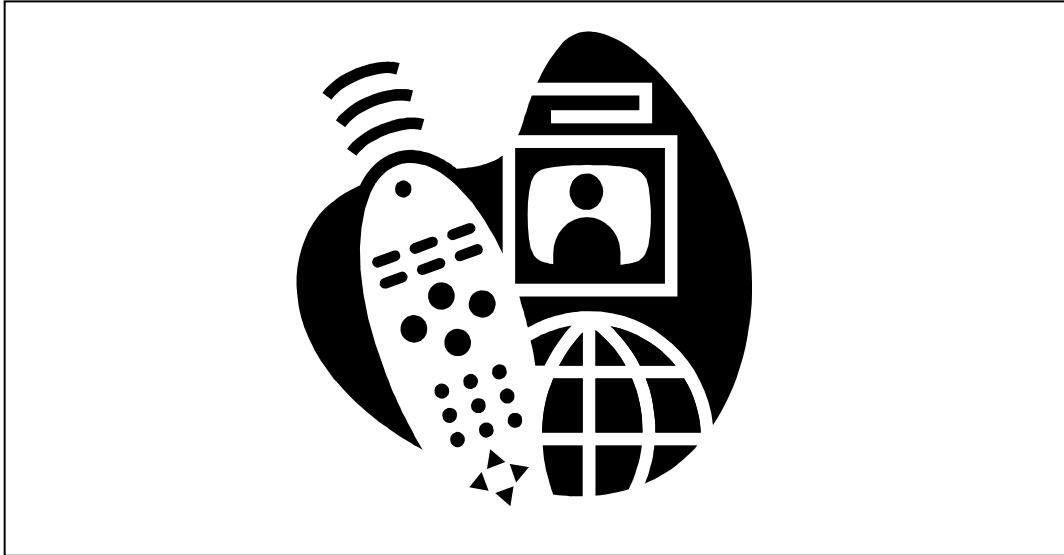


CHALMERS



Automatic DVB Performance Measurement Tool

Master of Science Thesis [Networks and Distributed Systems]

DAN YANG

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, July 2010

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet.

The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Automatic DVB Performance Measurement Tool

DAN YANG

© DAN YANG, July 2010.

Examiner: ARNE DAHLBERG

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

[Cover:

Clip art image copyright Microsoft Corporation, used with permission.

It is a combination of a set-top box, a TV set, a remote control and the Earth.]

Department of Computer Science and Engineering
Göteborg, Sweden July 2010

Abstract

Nowadays, there are several different standards that specify what performance should expect from a DVB (Digital Video Broadcasting) receiver, such as NorDig Unified, DTG D-Book and EICTA MBRAI. These standards are complicated and thousands of measurements are involved, so it would be very handy to repeat the measurements with an automatic tool. In this thesis work, only NorDig Unified 2.1 is concerned.

Since the test cases in NorDig Unified requirements for integrated receiver decoders are quite similar and repeatable, the work efficiency to fully evaluate the DVB performance could be improved by an automatic application. It would increase the test quality and decrease the manual labor a lot for each product.

Implementation of the tool is the most important part in this project. The application runs on a computer with a user friendly graphical interface. Its main task is to control the calibrated signal generator (Rohde & Schwarz SFU) and test the set-top boxes (STB) in Motorola product range. Besides, a database has been designed to store all these testing results, and analysis on the data has been presented as well.

In this project, the automatic DVB performance measurement tool is developed in Microsoft Visual C++ 6.0 with an ODBC (Open Database Connectivity) driver connecting to the database, and data presentation is done in Microsoft Excel 2003 using VBA (Visual Basic for Application).

Acknowledgement

I wish to express my deepest gratitude to Motorola for giving me the opportunity to accomplish this master thesis, to everyone at the company as well. Especially I would like to thank my supervisor Peter Karlén for the valuable support during the whole project, and it is a great fortune for me to meet him as my first boss.

Finally, I would like to thank my examiner Arne Dahlberg at Chalmers University of Technology for guidance and feedback.

Contents

1.	Introduction	1
1.1.	Background	1
1.2.	Problem description.....	1
1.3.	Purpose	1
1.4.	Method	1
1.5.	Structure	2
1.6.	Reading instructions	2
1.7.	Glossary.....	2
2.	Theoretical knowledge	4
2.1.	TCP/IP model.....	4
2.1.1.	Overview	4
2.1.2.	Architecture	4
2.1.3.	TCP/IP socket.....	5
2.2.	DVB-T.....	5
2.3.	DVB-C	6
2.4.	DVB-S.....	6
2.5.	IPTV	6
3.	IP based set-top boxes	7
3.1.	Overview	7
3.2.	Architecture	7
3.3.	Boot image	8
3.4.	Motorola products	8
4.	R&S® SFU broadcast test system.....	9
4.1.	Overview	9
4.2.	DVB-T environment settings	9
4.3.	DVB-C environment settings	11
4.4.	Remote control commands.....	11
4.4.1.	Basic commands.....	11
4.4.2.	DVB-T commands	11
4.4.3.	DVB-C commands	12
5.	Implementation.....	13
5.1.	Connection between PC and SFU	13
5.2.	Connection between PC and STB	14
5.3.	Graphical User Interface design.....	14
5.3.1.	Main thread	14
5.3.2.	Worker thread.....	16
5.4.	Test cases analysis.....	16
5.4.1.	DVB-T test cases.....	17
5.4.2.	DVB-C test cases	20
5.4.3.	Test case example.....	21
5.4.4.	Dual tuner problem.....	22
5.5.	Database design.....	22
5.5.1.	Entity-Relation diagram	22
5.5.2.	Database connection.....	23
5.5.3.	Operations on database.....	24
5.6.	Data Presentation.....	24
6.	Results	26

6.1.	Debug window	26
6.2.	Log file	27
6.3.	Database	28
6.4.	Report in Excel.....	29
7.	Conclusion.....	30
8.	Further development	31
9.	References	32
	Appendix	33

1. Introduction

1.1. Background

A set-top box (STB) is a simple computer specialized at decoding video data streams and displaying the content on a TV set. A particular kind of set-top box is called IP-STB; it can connect to a high speed IP-based network to receive TV streams instead of ordinary methods like terrestrial, cable and satellite broadcasting. By a two-way connection, operators can provide a wide range of services for users, such as advanced electronic program guide and video on demand, which encourages a high level of user interaction.

1.2. Problem description

In Europe, the DVB-C standard is used for cable TV broadcast, while the DVB-T standard is used for terrestrial TV broadcast. Motorola develops set-top boxes for IP-based infrastructures, but some products support terrestrial and cable TV infrastructures as well. The frontends (tuners) of DVB-C and DVB-T are complex RF-electronics which are very sensitive to noise. Besides, the software that controls these parts must be carefully tuned in order to get good performance. There are several different standards that specify what performance should expect from a DVB receiver, such as NorDig, DTG D-Book and EICTA MBRAI. These standards are complicated and involve thousands of measurements. It would become very handy to repeat these tests with an automatic DVB performance measurement tool.

1.3. Purpose

The purpose in this project is to improve the work efficiency to fully evaluate the DVB performance according to the NorDig Unified 2.1 standard. Since the test cases in NorDig are quite similar and repeatable, an automatic DVB performance measurement tool should increase the test quality and decrease the development time for each product.

1.4. Method

This work consists of literature study, implementation and result analysis.

The literature study focuses on NorDig Unified requirements for integrated receiver decoders and operating manual for Rohde & Schwarz SFU broadcast test system. NorDig Unified 2.1 specifies a set of equipment requirements for reception of DVB-based and related services from cable, satellite and terrestrial broadcast networks; in addition it includes requirements for reception via IP-based networks. The operating manual of SFU describes how to use this signal generator both locally and by remote control commands.

Implementation of DVB performance measurement tool is the most important part in this project, this application should run on a computer with a friendly graphical user interface. Its main task is to control the signal generator and test the products in Motorola product range. Besides, a database should be designed to store all testing results.

Results analysis mainly concentrates on presenting data in a suitable way and drawing conclusions from data performance.

1.5. Structure

Chapter 1, “Introduction”, gives a brief introduction and puts up the objectives of this thesis.

Chapter 2, “Theoretical knowledge”, explains the technical knowledge needed for this project.

Chapter 3, “Motorola set-top boxes”, describes different products in the test.

Chapter 4, “R&S SFU broadcast test system”, focuses on setting different arguments for DVB-T and DVB-C environments.

Chapter 5, “Implementation”, is the most important part in this report, including the measurement tool, database design and the report tool. It also offers solutions in details for each academic problem occurred during this period.

Chapter 6, “Results”, gives four different ways to present data.

Chapter 7, “Conclusion and further development”, contains guidelines on how to expand this measurement tool in the future.

Chapter 8, “References”, holds a list of the documents referenced to from the thesis.

1.6. Reading instructions

To get a brief overview of what this thesis report is about and what has been accomplished read:

Chapter 1, “Introduction”.

Chapter 6, “Results”.

If there is an interest for technical details of the project, read:

Chapter 4, “R&S SFU broadcast test system”.

Chapter 5, “Implementation”.

1.7. Glossary

BER

Bit Error Rate, it is the number of received bits that have been altered due to noise, interference and distortion, divided by the total number of transferred bits during a time interval.

DVB

Digital Video Broadcasting is a suite of internationally accepted open standards for digital televisions.

DVB-C

DVB Cable broadcast, it uses a fixed coaxial cable connection.

DVB-T

DVB Terrestrial broadcast, it is a traditional method of television broadcast signal delivery by radio waves.

DVB-S

DVB Satellite broadcast, it uses orbiting satellites.

GUI

Graphical User Interface, it is a type of user interface item that allows people to interact with programs in more ways than typing.

IP

Internet Protocol, it is the computer networking protocol used on the Internet.

IPTV

Internet Protocol Television, services are delivered using the architecture and networking methods of the Internet Protocol Suite over a packet-switched network infrastructure.

IRD

Integrated Receiver Decoder, it is an electronic device to pick-up a radio-frequency signal and convert digital information which is transmitted in it, which is the same as STB.

MPEG

Motion Picture Experts Group, it is the name of a collection of standards used for coding audio and visual information (like movies, video and music) in a compressed format. There are different revisions of the standard. The revisions are denoted by appending -1, -2, -3, or -4 to MPEG.

ODBC

Open Database Connectivity, it provides a standard software API for using database management systems (DBMS).

QEF

Quasi Error Free equals to a BER of $2.0e-4$ defined in NorDig Unified, which is a limit for users to get good TV performance.

STB

Set-Top Box, a multimedia device connected to a user's TV set that delivers analog or digital TV transmissions, music, games and other multimedia services to the user.

SFU

It is a broadcast test system, which can generate a DVB signal of any kind with option to adjust frequency, power level, noise, fading etc.

UCB

Un-Corrected Block. When the incoming BER is too high, error correction algorithms are not able to correct, and result in data loss. This is typically seen as video macroblocking defects.

2. Theoretical knowledge

2.1. TCP/IP model

2.1.1. Overview

The TCP/IP model, describes a set of guidelines and implementations of specific networking protocols to make computers communicate over a network. TCP/IP provides end-to-end connectivity, it specifies how data should be formatted, addressed, transmitted and received.

2.1.2. Architecture

The TCP/IP model consists of four layers. From lowest to highest, these are the link layer, the Internet layer, the transport layer and the application layer [8] as shown in figure 1.

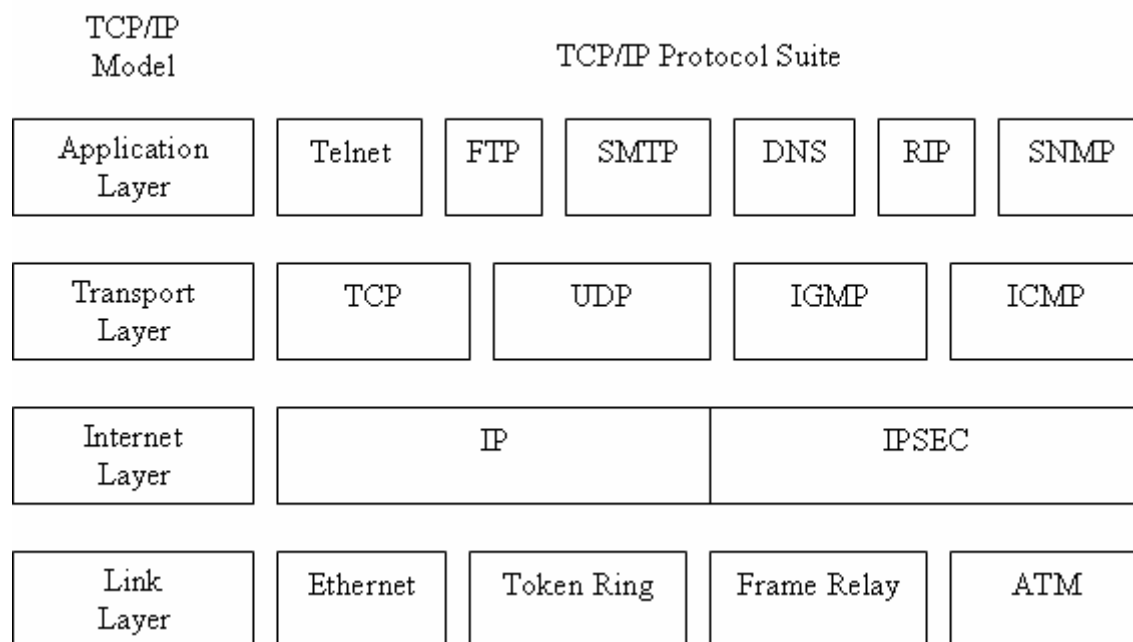


Figure 1. The TCP/IP architecture and the other protocols in the TCP/IP suite [1]

Application layer consists of application programs and user interfaces. In practice it sends unbroken data stream to the transport layer. It supports a number of protocols such as SMTP, FTP and Telnet.

Transport layer provides data integrity and a highly reliable communication service. This layer accepts information to be transmitted as a stream, and it also returns information to the recipient as a stream. The main protocols that are used in this layer include TCP, UDP and RTP.

Internet layer is responsible for routing messages through internet. Here, data is packaged into IP datagram, which contain the address of source and destination and checksum value. If the receiver detects a transmission error by using the checksum value of the datagram, it simply ignores the datagram without notifying the higher-layer entity. The main protocols used in this layer are IP, ICMP, ARP as well as RARP.

Link layer specifies how to send data through the network physically. Unlike higher level protocols, this layer must understand the details of the underlying physical network, such as the packet structure, maximum frame size and the physical address scheme. This makes the data transmitted across the network correctly. Typical examples of protocols used throughout this layer are the "Token Ring", "Ethernet" and FDDI.

2.1.3. TCP/IP socket

The IP layer sends information in small chunks of data called packets. These packets have the receiver’s IP address in the header. The infrastructure of the network such as routers, switches forwards the packet to the receiver one step further. In the IP layer, there is no guarantee that the packet has been transmitted successfully or in order, so it needs other mechanism to control the transmission – TCP layer.

The TCP layer makes all packets arrive at the receiver in the same order as they were sent. There is a unique id for each packet, and this enables reconstruction at the receiver’s side. Each received message is confirmed with an acknowledgment (ACK). If a packet is lost, either the original packet or its ACK, the sender will notice it with a timeout and invoke the IP layer to resend the packet.

Once a TCP/IP socket is created, it is associated with a port. This port is used together with the IP address to start the connection [8].

2.2. DVB-T

DVB-T is an abbreviation for Digital Video Broadcasting-Terrestrial, which is the DVB European consortium standard for the broadcast transmission of digital terrestrial television. It transmits compressed digital audio, video and other data in an MPEG transport stream by COFDM modulation.

Table 1. Parameter values for DVB-T environment

Arguments	Value
Carrier	2K, 8K
Modulation	QPSK, 16 QAM, 64 QAM
Channel bandwidth	6 MHz, 7 MHz, 8 MHz
Guard interval	1/4, 1/8, 1/16, 1/32
Code rate	1/2, 2/3, 3/4, 5/6, 7/8

2.3. DVB-C

DVB-C stands for Digital Video Broadcasting-Cable and is the DVB European consortium standard for the broadcast transmission of digital television over cable. It transmits an MPEG-2 or MPEG-4 family digital audio/video stream by a QAM modulation with channel coding.

Table2. Parameter values for DVB-C environment

Arguments	Value
Symbol rate	0.1 MS/s to 8 MS/s
Modulation	16 QAM, 32 QAM, 64 QAM, 128 QAM, 256 QAM

2.4. DVB-S

DVB-S stands for Digital Video Broadcasting–Satellite. It is suitable to use on different satellite transponder bandwidths, and is compatible with MPEG-2 coded TV services. Flexibility defined within the specification enables the transmission capacity to be used for a variety of TV service configurations.

2.5. IPTV

Internet Protocol television (IPTV) is a system, and internet television services are delivered by the architecture and networking methods of the Internet Protocol Suite over a packet-switched network infrastructure.

IPTV services may be classified into three main groups: live television, time-shifted programming and video on demand (VOD). Time-shifted programming makes users record the TV programs and do other operations as fast forward, rewind, pause, etc. Video on demand allows users to select and watch/listen to video/audio content as they like.

3. IP based set-top boxes

3.1. Overview

An IP based set-top box is a specialized computer which can decode audio/video transmissions sent via an IP based network and provide interaction to users. Each STB is connected to a high speed network, and servers in this network provide the STB information. To decrease the demands on the network’s bandwidth, multicast can be used to broadcast TV channels. By doing this, each TV channel needs to be sent out only once no matter how many users are watching the channel and each channel has its own multicast address that the STBs can listen.

A typical STB contains a CPU, a RAM, a flash memory and an MPEG decoder. The MPEG decoder is used to decompress digital TV transmissions, since the decompression is a complicated procedure, it is much better to have a hardware dedicated to decode rather than a fast CPU, then the main CPU only deals with GUI, networking and other simpler tasks, where a low cost one can be used.

3.2. Architecture

The STBs used in this project run on a GNU/Linux operating system, which is open source and offers a familiar programming environment. The main layers of the STB architecture are shown in the figure 2.

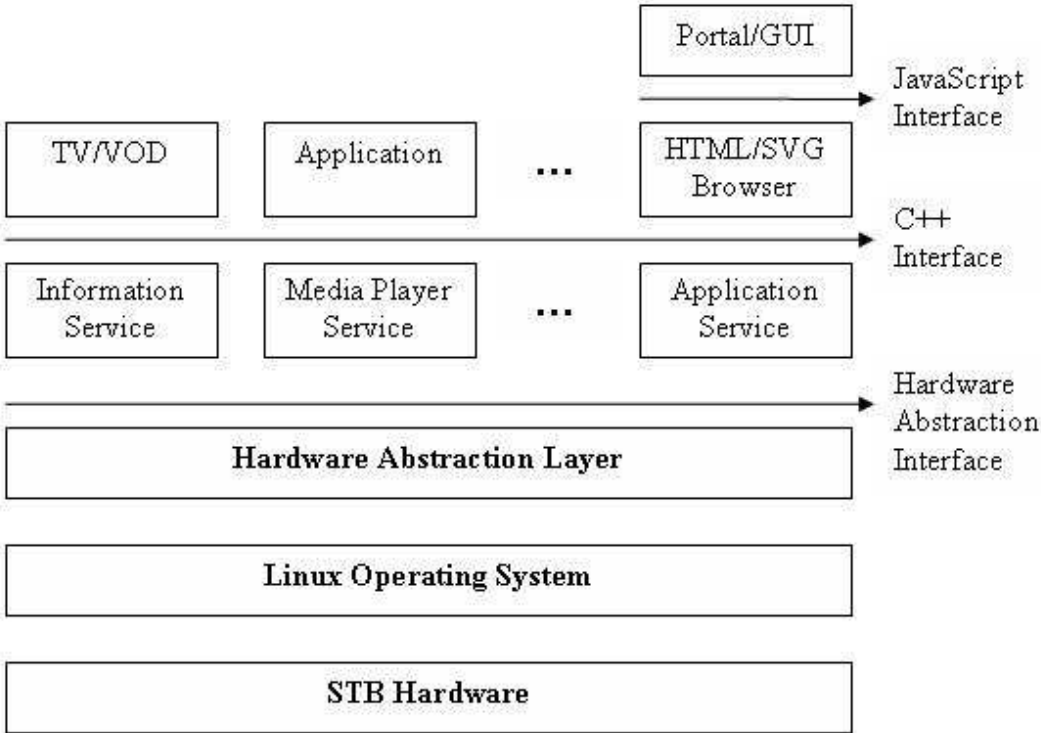


Figure 2. The architecture of the STB

3.3. Boot image

A boot image is a file that contains all software, and is loaded into the STB when it boots up. The boot image is distributed to all STBs by a multicast server. This allows a scalable system where it takes as long to send information to one STB as to two or even 1000 STBs. A boot image can be stored in the STB or downloaded when the STB powers on. In this project, the boot image is downloaded every time when the STB boots up, this makes it much easier to change a boot image.

3.4. Motorola products

There are many different types of STBs in Motorola product range. The STBs to be tested in this project are VIP1903T, VIP1963T, VIP1903C and VIP1963C. All these STBs include Ethernet interface and DVB-T or DVB-C frontends. Table 3 shows the different arguments for the different STBs. Figure 3 is a picture of VIP1963T.

Table3. Parameters for Motorola products

Product	Product type	Tuner type	Hard disk
VIP1903T	DVB-T	Single Tuner	No
VIP1963T	DVB-T	Dual Tuner	Yes
VIP1903C	DVB-C	Single Tuner	No
VIP1963C	DVB-C	Dual Tuner	Yes



Figure 3. Motorola VIP1963T STB [2]

4. R&S® SFU broadcast test system

4.1. Overview

The R&S® SFU is a test system which offers solutions for broadcast and mobile TV standards. It has been designed as a platform for various applications, and is open for future options. It provides easy remote access by general purpose interface bus (GPIB) and local area network (LAN). Figure 4 is a picture in its front view.

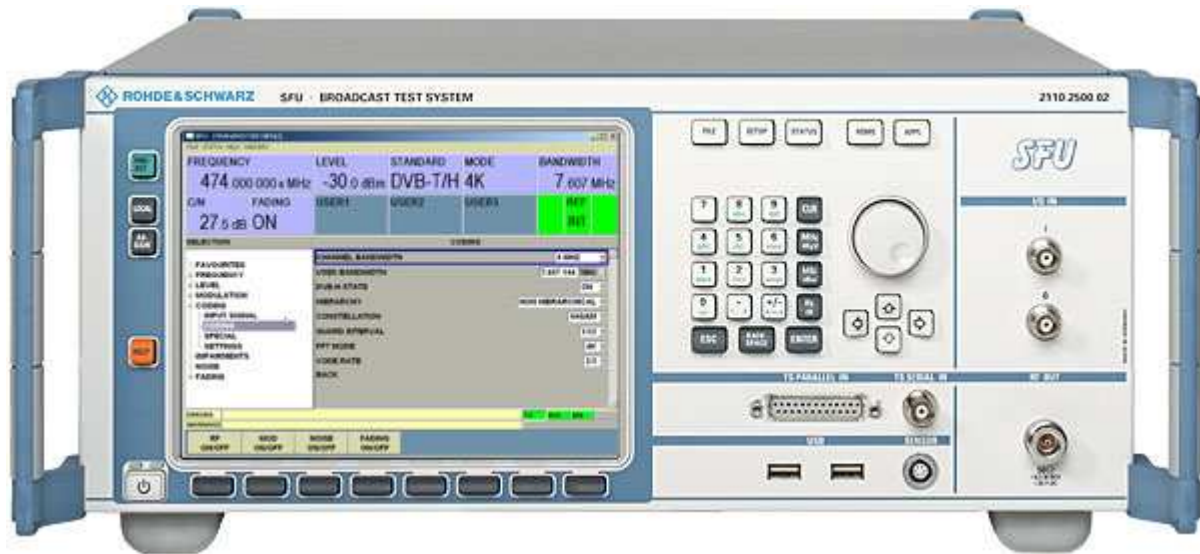


Figure 4. The front view of the SFU [3]

4.2. DVB-T environment settings

When testing a DVB-T set-top box, the standard should be set DVB-T/H. According to the different test cases, select the right option in different fields, frequency, power level, power offset, FFT mode, channel bandwidth, constellation, guard interval, code rate, noise, fading and etc.

Frequencies are divided into very high frequency (VHF) and ultra high frequency (UHF). VHF starts from 30 MHz to 300 MHz, while UHF has a range from 300 MHz to 3 GHz. When testing VHF cases, choose channel bandwidth 7 MHz and select 8 MHz for UHF cases.

FFT mode is short for fast fourier transform mode, which can be 2K or 8K.

Constellation, it can be set to QPSK, 16 QAM or 64 QAM.

Phase-shift keying (PSK) is a digital modulation scheme that conveys data by changing the phase of a reference signal. QPSK, Quadrature Phase-Shift Keying, uses four points on the constellation diagram equispaced around a circle [9]. Each adjacent symbol only differs by one bit just as the figure 5 shows.

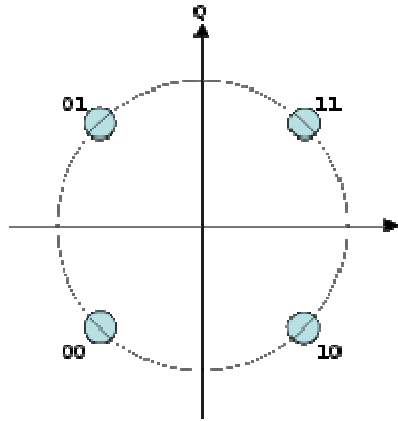


Figure 5. Constellation for QPSK [5]

QAM, Quadrature Amplitude Modulation, it is both an analog and a digital modulation scheme. It conveys two analog message signals, or two digital bit streams, by changing the amplitudes of two carrier waves, using the amplitude-shift keying (ASK) digital modulation scheme or amplitude modulation (AM) analog modulation scheme [9]. Figure 6 is an example for 16 QAM.

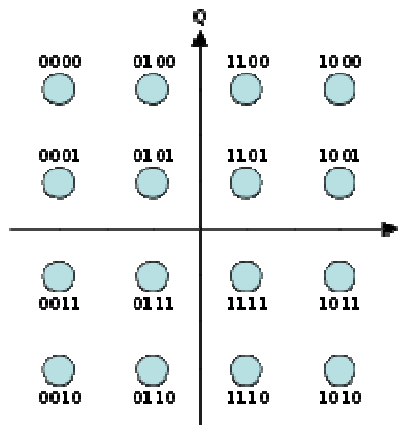


Figure 6. Constellation for 16 QAM [6]

Guard interval is used to ensure that distinct transmissions do not interfere with one another, and it can be set to 1/4, 1/8, 1/16, 1/32. Longer guard period allows more distant echoes to be tolerated, but it reduces the channel efficiency. For example, 1/32 gives lowest protection but the highest data rate. 1/4 results in the best protection but the lowest data rate.

Code rate, it is typically a fractional number. If the code rate is k/n , for every k bits of useful information, the coder generates totally n bits of data, of which $n-k$ are redundant. For example, The code rate may typically be 1/2, 2/3, 3/4, 5/6, 7/8, corresponding to that one redundant bit is inserted after every single, second, third, fifth, seventh bit.

Noise, when it is on, the C/N value should be set as well.

C/N, Carrier to Noise ratio, is defined as the ratio of the received modulated carrier signal power C to the received noise power N after the receive filters.

$$CNR = \frac{C}{N}$$

C/N ratio is often specified in decibels (dB):

$$CNR_{dB} = 10 \log_{10} \left(\frac{C}{N} \right) = C_{dBm} - N_{dBm}$$

Fading, the R&S SFU allows the user to superimpose fading on the baseband signal at the output of various baseband blocks in real-time. In this project, 0 dB echo channel is the only case to simulate, details about this setting is described in 4.4.2.

4.3. DVB-C environment settings

In DVB-C environment, the standard should be set to DVB-C. According to the different test cases, select the right option in different fields, frequency, power level, power offset, noise, constellation, symbol rate and etc.

Constellation, it can be set to 16 QAM, 32 QAM, 64 QAM, 128 QAM or 256 QAM.

Symbol rate, is the number of symbol changes made to the transmission medium per second. It is measured in baud (Bd) or symbols/second. It can be set in the range 0.1 MS/s to 8 MS/s with a resolution of 1 Hz.

4.4. Remote control commands

The SFU can be controlled by remote commands, and the most frequently used commands are listed here.

4.4.1. Basic commands

Frequency:	FREQ 474000000
Power level:	POW -30
Power level offset:	POW:OFFS -2
Standard:	DM:TRAN DVBT DVBC
Noise:	NOIS OFF ADD ONLY
C/N:	NOIS:CN 20

4.4.2. DVB-T commands

Channel bandwidth:	DVBT:CHAN:BAND BW_7 BW_8
Constellation:	DVBT:CONS T4 T16 T64
FFT mode:	DVBT:FFT:MODE M2K M8K
Guard interval:	DVBT:GUAR:INT G1_4 G1_8 G1_16 G1_32
Code rate:	DVBT:RATE R1_2 R2_3 R3_4 R5_6 R7_8
Fading state:	FSIM ON OFF
Fading path state:	FSIM:DEL:GRO1:PATH1:STAT ON OFF
Fading path profile:	FSIM:DEL:GRO1:PATH1:PROF SPAT PDOP RAYL RICE
Basic delay:	FSIM:DEL:GRO2:BDEL 1.95e-6
Additional delay:	FSIM:DEL:GRO1:PATH1:ADEL 0
Speed for moving receiver:	FSIM:DEL:GRO1:PATH1:SPE 0

0 dB echo channel setting: FSIM:DEL:GRO1:PATH1:STAT ON
FSIM:DEL:GRO1:PATH1:PROF SPAT
FSIM:DEL:GRO1:PATH1:SPE 0
FSIM:DEL:GRO1:PATH1:ADEL 0
FSIM:DEL:GRO2:PATH1:STAT ON
FSIM:DEL:GRO2:PATH1:PROF SPAT
FSIM:DEL:GRO2:PATH1:SPE 0
FSIM:DEL:GRO2:PATH1:ADEL 0
FSIM:DEL:GRO2:BDEL 1.95e-6

4.4.3. DVB-C commands

Constellation: DVBC:CONS C16|C32|C64|C128|C256
Symbol rate: DVBC:SYMB 6.9e6

5. Implementation

This project includes five modules, as figure 7 shows. The central part is the windows PC application called DVB measurement tool, which is used to connect the SFU and the STB by socket communication. The main procedure of testing is like this; the DVB measurement tool loads the test session in the beginning, then sets arguments in the signal generator (SFU) by remote control commands. After the environment has been simulated, the tool makes the STB tune to a certain frequency and records the stable data from the STB, which focuses on uncorrected block (UCB) and bit rate error (BER). During the recording period, the application displays debug information in real time and saves the results both in the log file (*.txt file) and the database. When data collection is done, the report tool gives out the performance of each test case by analyzing the results in the database.

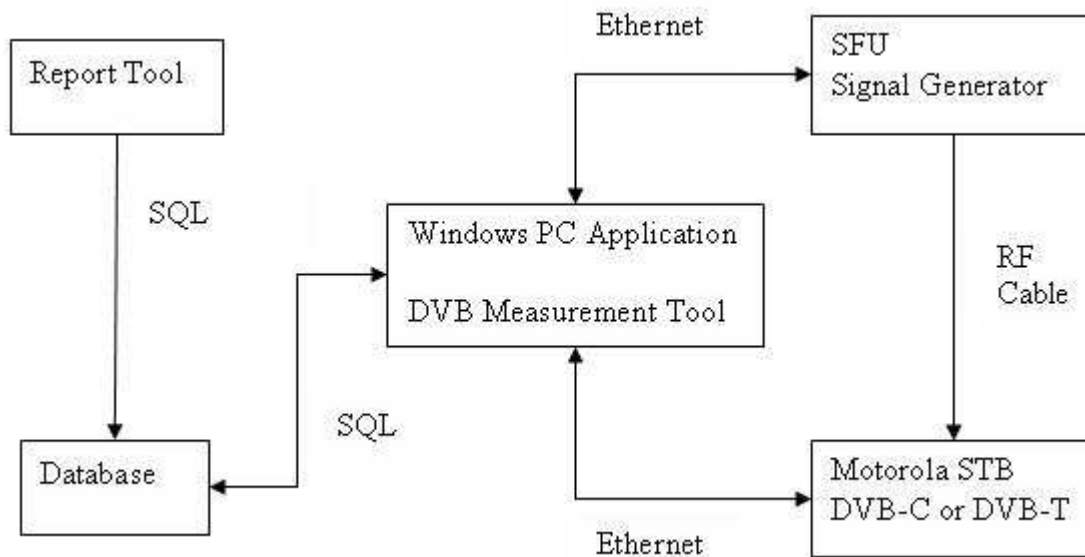


Figure 7. Modules for DVB measurement tool

From the programming point of view, there are five classes, which are called SFU, STB, Database, CDVBTOOLDlg and CWorkerThread. The functions of each class are detailed described in the next few pages.

5.1. Connection between PC and SFU

The SFU class handles each operation related with the SFU, including initializing the connection between the computer and the SFU, setting signal generator to a correct environment and getting the status of different fields in the SFU.

Building a connection between the PC and the SFU relies on socket communication. A static IP address is assigned for the SFU, and 5025 is the port number for remote control as the SFU manual has defined. After executing a connect function provided by the C++ library, the computer and the SFU have built the connection.

The format of the connect function is like this.

```
int connect (SOCKET s, const struct sockaddr *name, int namelen); [13]
```

Sending and receiving strings are done by functions provided by the C++ library as well. The formats of the two functions are as below.

```
int send (SOCKET s, const char *buf, int len, int flags); [13]
```

```
int recv (SOCKET s, char *buf, int len, int flags ); [13]
```

Setting the SFU arguments and getting status of the SFU are simply done by sending remote commands which are specified in Chapter 4.4. with details. For example, setting the power offset -5.7dBm is by sending the string “POW:OFFS -5.7” to the SFU, and getting the status of the power offset is by sending the string “POW:OFFS?”, replacing ‘ -5.7’ with ‘?’.

The main methods of the SFU class are listed in the appendix.

5.2. Connection between PC and STB

The STB class deals with every operation with the STB, includes initializing the connection between the computer and the STB, tuning the STB and getting stable quality of the STB.

Making a connection between the PC and the STB also relies on socket communication. A static IP address is assigned for STB, and the port number 23 is used for remote control.

There are two tuning modes for STB, DVB-T and DVB-C. The arguments in DVB-T are frequency, tuner interface, channel bandwidth. The arguments in DVB-C are frequency, tuner interface, modulation and symbol rate. Two methods in STB class handle two tuning procedures respectively.

The formats of the two tuning commands are like this.

```
run.sh -t [type] -f [frequency] -i [instance] -b [bandwidth]
```

```
run.sh -t [type] -f [frequency] -i [instance] -m [modulation] -s [symbol rate]
```

After tuning, the STB measures the quality of signal including UCB, BER and other results. Since UCB and BER are the most important, there are methods to extract them respectively. Besides, the software in the STB that measures the quality outputs the result every second and it needs some time to become stable, so a period of 26 seconds is chosen when testing.

The methods of the STB class are listed in the appendix.

5.3. Graphical User Interface design

Two threads are used for GUI, the main thread deals with all display operations, while the worker thread handles all the measurements on the STB. They communicate with each other through message passing. The main thread makes the worker thread start or stop testing, while the worker thread does the real test procedure and feedbacks both the results and current progress to the main thread.

5.3.1. Main thread

The GUI for this project is shown in figure 8, which is implemented by MFC. It contains three parts: test control, test information and STB information.

In the Test Control part, there is a combo box for users to select different test sessions, two buttons to start and stop testing. Once a test session is selected, the Start button is enabled. When Start is clicked, it turns to grey which means you can not enable the button unless the test completes or stopped by the user.

In the STB Info part, there are six edit boxes for users to type in the information of different STBs. The information is saved both in log file and database. The Debug window is a read-only area; it outputs the real time results.

In the Test Info part, there are four read-only edit boxes. When Start button is clicked, the status becomes running. When the test is completed, it changes to completed. When the Stop button is clicked, the status turns to stopping, once the test is totally stopped, it sets to stopped. Start/Completion time is the time when the test starts/completes. Est. Duration is the period that comes from a lot of old tests, so it seems accurate somehow. Each test session has several test cases; the Test Session Progress is the progress for the whole test session, while the Test Case Progress is the progress for each test case. From the two progress bars and Est. duration, it is obvious to see which step the test is running and when it is estimated to finish.

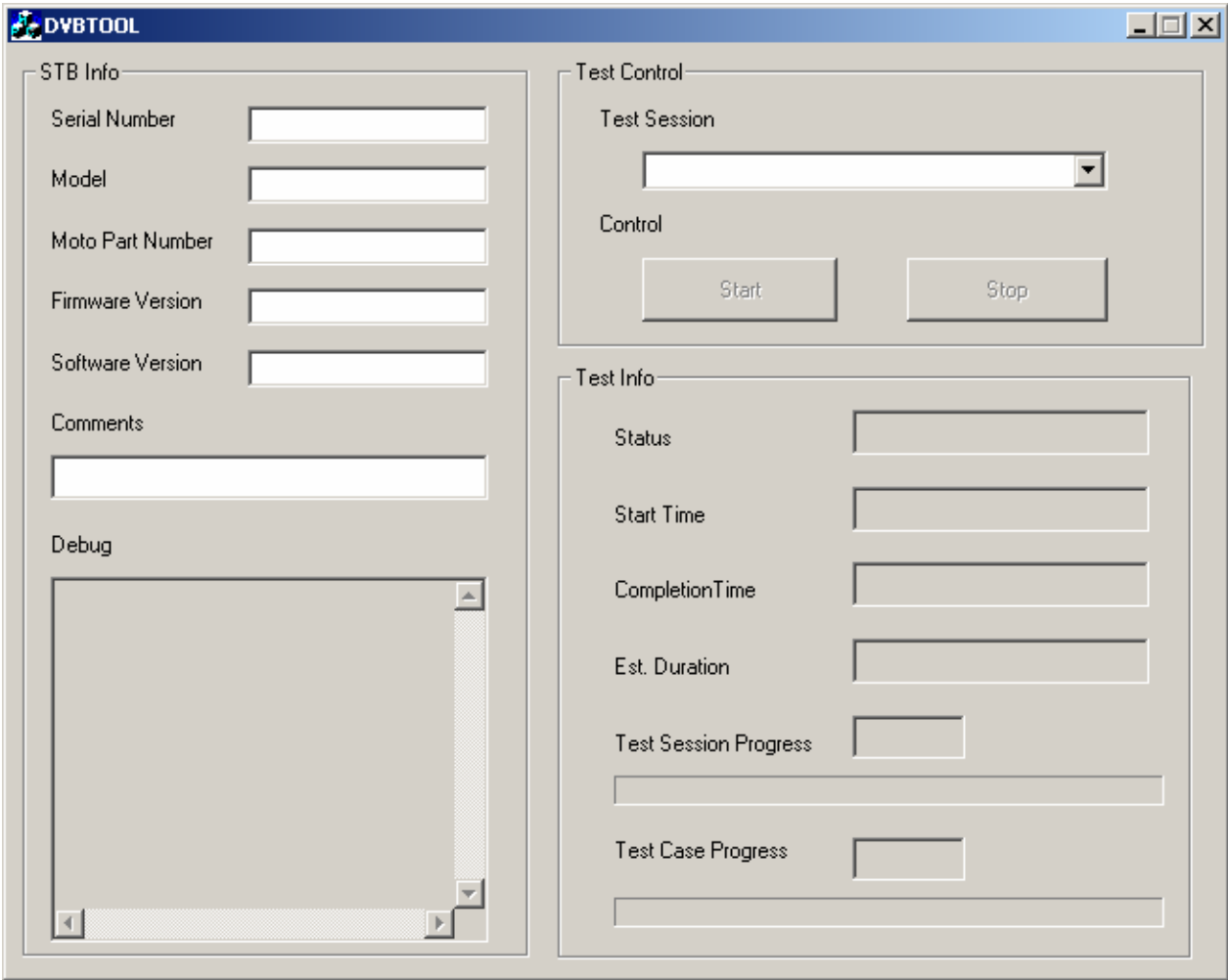


Figure 8. The GUI for DVB measurement tool

When this tool begins to run, it loads the initialized file (*.ini file) to find the configurations for the different devices. Here is the content listed in the initialized file.

```
[Configuration]
LOG_DIR=D:\Dan\working\DVBTOOL\LOG\
SESSION_DIR=D:\Dan\working\DVBTOOL\SESSIONS
DEBUG_LEVEL=1
[SFU]
SFU_IP=192.168.1.84
SFU_PORT=5025
SFU_PAD_LOSS=-5.7
[STB]
STB_IP=192.168.0.20
[DB]
DB_USE=1
DB_SERVER=server.com
DB_PORT=3306
DB_SOURCE=dvbtool
DB_USER=myuser
DB_PASSWORD=mypass
DB_DRIVER=MySQL ODBC 5.1 Driver
```

In the Configuration part, it contains the directories of the logs and the test session files. DEBUG_LEVEL is the argument which controls the output content in the log. When it equals to 1, both the debug information and the results are written in the log. When it equals to 0, only the results are saved.

In the SFU part, it contains the IP address, the port number and the insertion loss of the matching pad used.

In the STB part, the IP address of the STB is listed here. For the Motorola products in this project, the IP address is always the same.

In the DB part, DB_USE is the argument which determines whether a database is used or not. When it equals to 1, it uses the database. The remaining arguments are used to open connection, including the server name, the port number, the source name, the user id, the password and the version of the database driver.

5.3.2. Worker thread

The worker thread does the test when the main thread asks it to do, and it handles both DVB-T and DVB-C tests. First it reads the test session file (*.ses file) to find out which cases will be tested, and then stores the arguments needed for each test case. There is an algorithm designed for every test case. When running these algorithms, the results are shown in the debug window and saved in both the log file and the database as well. The worker thread can be interrupted by the main thread with Stop button clicked.

5.4. Test cases analysis

There are nine test cases in DVB-T and four cases in DVB-C totally. The user can either run the full test or partial test by modifying the test session file (*.ses file). These test cases are derived from NorDig Unified 2.1 [7], which specifies a set of equipment requirements for

reception of DVB-based and related services from cable, satellite, terrestrial broadcast networks and IP-based networks.

5.4.1. DVB-T test cases

Test case 1: Center Frequencies

Purpose:

For supported frequency ranges, the frontend shall be able to tune to the center frequency f_c of the incoming DVB-T RF signal.

Description:

Change frequency to different values, and check the sum of UCB during a certain period.

Test parameters:

Power level: -60 dBm

Transmission mode: 8K

Constellation: 64 QAM

Code rate: 2/3

Guard interval: 1/8

Frequency: f_c in the formula

7 MHz bandwidth: $f_c = (107.5 + L * 7)$ MHz, where L is an integer number from 0 to 27.

8 MHz bandwidth: $f_c = (114 + K * 8)$ MHz, where K is an integer number from 0 to 93.

Test case 2: Frequency Offset

Purpose:

The NorDig IRD should be able to receive signals with an offset of 50 kHz from the normal frequency.

Description:

Apply the first and the last frequencies to the tuner, test the sum of UCB. Then shift the frequency from its normal value by +/- 50 KHz, test the sum of UCB again.

Test parameters:

Power level: -60 dBm

Transmission mode: 8K

Constellation: 64 QAM

Code rate: 2/3

Guard interval: 1/8

Frequency: 177.5 MHz and 226.5 MHz in VHF band

474 MHz and 858MHz in UHF band

Test case 3: Signal Bandwidths

Purpose:

For supported frequency rangers, the NorDig IRD which could receive 7 MHz signals should be able to receive 8 MHz signals as well. If 8 MHz bandwidth is supported, it shall automatically detect which DVB-T signal bandwidth is being used and it shall be possible to receive the 8 MHz signals on the 7 MHz VHF channel frequencies.

Description:

For a special VHF band frequency, both 7 MHz bandwidth and 8 MHz bandwidth are applied to the demodulator, and the sum of UCB is measured respectively.

Test parameters:

Power level:	-60 dBm
Transmission mode:	8K
Constellation:	64 QAM
Code rate:	2/3
Guard interval:	1/8
Frequency:	198.5 MHz both at 7 MHz and 8 MHz bandwidth.

Test case 4: Modes

Purpose:

The NorDig IRD should be able to correctly demodulate all non-hierarchical modes with any combination of constellation (QPSK, 16 QAM, 64 QAM), code rate (1/2, 2/3, 3/4, 5/6, 7/8), guard interval (1/4, 1/8, 1/16, 1/32) and transmission mode (2K, 8K).

Description:

A mode is a combination of constellation, code, guard interval and transmission mode. For each mode, 8 MHz signal is applied to the demodulator, and the sum of UCB is measured.

Test parameters:

Frequency:	666 MHz
Power level:	-60 dBm
Transmission mode:	2K, 8K
Constellation:	QPSK, 16 QAM, 64 QAM
Code rate:	1/2, 2/3, 3/4, 5/6, 7/8
Guard interval:	1/4, 1/8, 1/16, 1/32

Test case 5: C/N Performance on Gaussian Channel

Purpose:

QEF is a constant with a value $2.0e-4$, which is a limit for users to get good TV performance. When bit error rate (BER) reaches QEF, it reaches a limit of good performance. The NorDig IRD shall have at least the QEF performance for the C/N ratios.

Description:

For each constellation and code rate, the C/N ratio is measured to reach QEF. All frequencies in both VHF and UHF bandwidth with different modulation and different code rate are tested.

Test parameters:

Power level:	-50 dBm
Transmission mode:	8K
Guard interval:	1/4
Constellation:	QPSK, 16 QAM, 64 QAM
Code rate:	1/2, 2/3, 3/4, 5/6, 7/8
Frequency:	177.5 MHz to 226.5 MHz in VHF band 474 MHz to 858 MHz in UHF band

Test case 6: C/N Performance on 0dB Echo Channel

Purpose:

It is the same purpose as the test case 5.

Description:

This test is performed in various DVB-T modes with 0dB echo (delay 1.95 us), and the C/N ratio is measured to reach QEF.

Test parameters:

Power level:	-50 dBm
Transmission mode:	8K
Guard interval:	1/4
Constellation:	QPSK, 16 QAM, 64 QAM
Code rate:	1/2, 2/3, 3/4
Frequency:	198.5 MHz in 7 MHz bandwidth 666 MHz in 8 MHz bandwidth

Test case 7: Minimum Input Level on Gaussian Channel

Purpose:

The NorDig IRD shall provide QEF reception for the minimum signal levels in the supported frequency ranges.

Description:

Measure the minimum input level without noise needed to reach QEF.

Test parameters:

Transmission mode:	8K
Guard interval:	1/4
Constellation:	QPSK, 16 QAM, 64 QAM
Code rate:	1/2, 2/3, 3/4, 5/6, 7/8
Frequency:	177.5 MHz to 226.5 MHz in VHF band 474 MHz to 858 MHz in UHF band

Test case 8: Minimum Input Level on 0dB Echo Channel

Purpose:

It is the same purpose as the test case 7.

Description:

Measure the minimum input level needed to reach QEF on the 0dB echo profile without noise. Vary the echo delay from 1.95 us to 212 us.

Test parameters:

Transmission mode:	8K
Guard interval:	1/4
Constellation:	QPSK, 16 QAM, 64 QAM
Code rate:	1/2, 2/3, 3/4, 5/6, 7/8
Delay:	1.95 us, 10 us, 28 us, 56 us, 70 us 90 us, 105 us, 112.1 us, 170 us, 212 us
Frequency:	198.5 MHz in 7 MHz bandwidth 666 MHz in 8 MHz bandwidth

Test case 9: Maximum Receiver Signal Input Level

Purpose:

The receiver should provide QEF reception for DVB-T signals up to a power level -35 dBm.

Description:

For the stated DVB-T modes, verify that the QEF and UCB at different power levels.

Test parameters:

Frequency:	666 MHz
Transmission mode:	8K
Guard interval:	1/4, 1/8
Constellation:	64 QAM
Code rate:	2/3, 3/4
Power level:	-35 dBm, -20 dBm, -10 dBm

5.4.2. DVB-C test cases

Test case 1: QEF Performance versus Frequency

Purpose:

The NorDig IRD shall have QEF performance at all defined frequencies.

Description:

Determine the minimum C/N for QEF performance at all listed frequencies.

Test parameters:

Power level:	-40 dBm
Constellation:	16 QAM, 64 QAM, 128 QAM, 256 QAM
Symbol rate:	6.9 Mbaud
Frequency:	106 MHz to 858 MHz

Test case 2: Minimum Input Level with NorDig Noise

Purpose:

The NorDig IRD shall be able to handle DVB-C signals at low power levels.

Description:

Apply a signal with NorDig noise; find the minimum input level needed for QEF performance at all frequencies and in different modes.

Test parameters:

Constellation:	16 QAM, 64 QAM, 128 QAM
Symbol rate:	6.9 Mbaud
Nodig Noise (C/N):	20.5 dB, 26.5 dB, 29.5 dBm
Frequency:	106 MHz to 858 MHz

Test case 3: QEF Performance versus Symbol Rate

Purpose:

The NorDig IRD shall be able to handle DVB-C signals at symbol rates between 3.0 and 7.0Mbaud, meanwhile the QEF performance shall be met.

Description:

Find lowest C/N to meet QEF performance with minimum and maximum power level.

Test parameters:

Frequency:	666 MHz
Power level:	-38.8 dBm, -61.8 dBm
Constellation:	16 QAM, 32 QAM, 64 QAM, 128 QAM, 256 QAM
Symbol rate:	3.0 Mbaud, 3.5 Mbaud, 4.0 Mbaud, 4.5 Mbaud, 5.0 Mbaud, 5.5 Mbaud, 6.0 Mbaud, 6.9 Mbaud

Test case 4: BER versus C/N

Purpose:

The NorDig IRD shall have a BER performance better than $2e-4$ for the C/N ratios specified.

Description:

For each one of the constellations and symbol rates, vary C/N and record BER values.

Test parameters:

Frequency:	666 MHz
Power level:	-40 dBm
Constellation:	16 QAM, 32 QAM, 64 QAM, 128 QAM, 256 QAM
Symbol rate:	3.0 Mbaud, 6.9 Mbaud

5.4.3. Test case example

The test case 7 in DVB-T is used as an example and part of the test session file (*.ses file) is as follows.

```
TESTCASE7=TRUE
DESC7=Minimum Input Level on Gaussian Channel
TUNER7=SINGLE
CONSTELL7=T4
#CONSTELL7=T16
CONSTELL7=T64
CODERATE7=R1_2
#CODERATE7=R2_3
CODERATE7=R3_4
CODERATE7=R5_6
CODERATE7=R7_8
FREQ7VHF=177500000
#FREQ7VHF=198500000
FREQ7VHF=226500000
#FREQ7UHF=474000000
FREQ7UHF=522000000
#FREQ7UHF=570000000
FREQ7UHF=618000000
FREQ7UHF=666000000
#FREQ7UHF=714000000
FREQ7UHF=762000000
FREQ7UHF=810000000
FREQ7UHF=858000000
```

With this file, it is quite convenient to configure the test. The line is neglected if there is a “#” character in front of the key word. In this case, the system will run through all combinations of listed constellations, code rates and frequencies. The example will result in 64 ($2*4*8$) combinations of measurements.

5.4.4. Dual tuner problem

Since some of the set top boxes have two tuners, one tuner will infect the other when both tuners work on similar frequency. This is called a dual tuner problem. In order to solve this, make one tuner stay at a frequency that the other will never use, and then do the test just the same as the single tuner. For example, there are tuner 1 and 2 in the same STB; tuner 1 is tuned to frequency 462 MHz, which is never used during the whole test. Then stop the tuning procedure, tuner 1 will stay at 462 MHz. After that, do the test on tuner 2.

5.5. Database design

5.5.1. Entity-Relation diagram

In order to keep data safe, a database has been designed. As shown in figure 9, there are three entities in this database, TestSession, TestCase and Result. The primary key of each part is StartedTime, CaseId and CurrentTime. Foreign keys of result are StartedTime in TestSession and CaseId in TestCase. The relations between these entities are quite obvious; a TestSession has many Results, and such Results correspond to one TestCase.

In the TestSession entity, StatedTime and SessionName are not null, while Model, Comment, SerialNumber, MotoPartNumber, FirmwareVersion and SoftwareVersion can be filled in by the user before clicking Start button; all these arguments are inserted into database at the beginning of the test. When the program completes correctly, there will be a string formatted time value in CompletedTime. If the program is stopped by the user, a value will be filled in StoppedTime. These two values are updated into database until the test is finished or stopped.

In the TestCase entity, data in this table is inserted in advance, because each test case is predefined, and no operations are done on these tables during testing period.

In the Result entity, data is inserted into the table once there comes an item of result. From the ER diagram, it is clear to see that there are four kinds of attributes, including common attributes, DVB-T attributes, DVB-C attributes and STB attributes. Common attributes contain CurrentTime, Frequency, PowerLevel and CN (Carrier to Noise ratio). STB attributes have UCB, BER and Tuner (tuner instance, 0 for single tuner, 0 for the first instance of dual tuner, 1 for the second instance of dual tuner). DVB-T attributes consists of ConstellationT, ChannelBandwidth, FFTMode, CodeRate, GuardInterval and Delay. DVB-C attributes are made up of ConstellationC and SymbolRate. For every item of Result, common attributes and STB attributes are not null, either DVB-T or DVB-C attributes are not null as well.

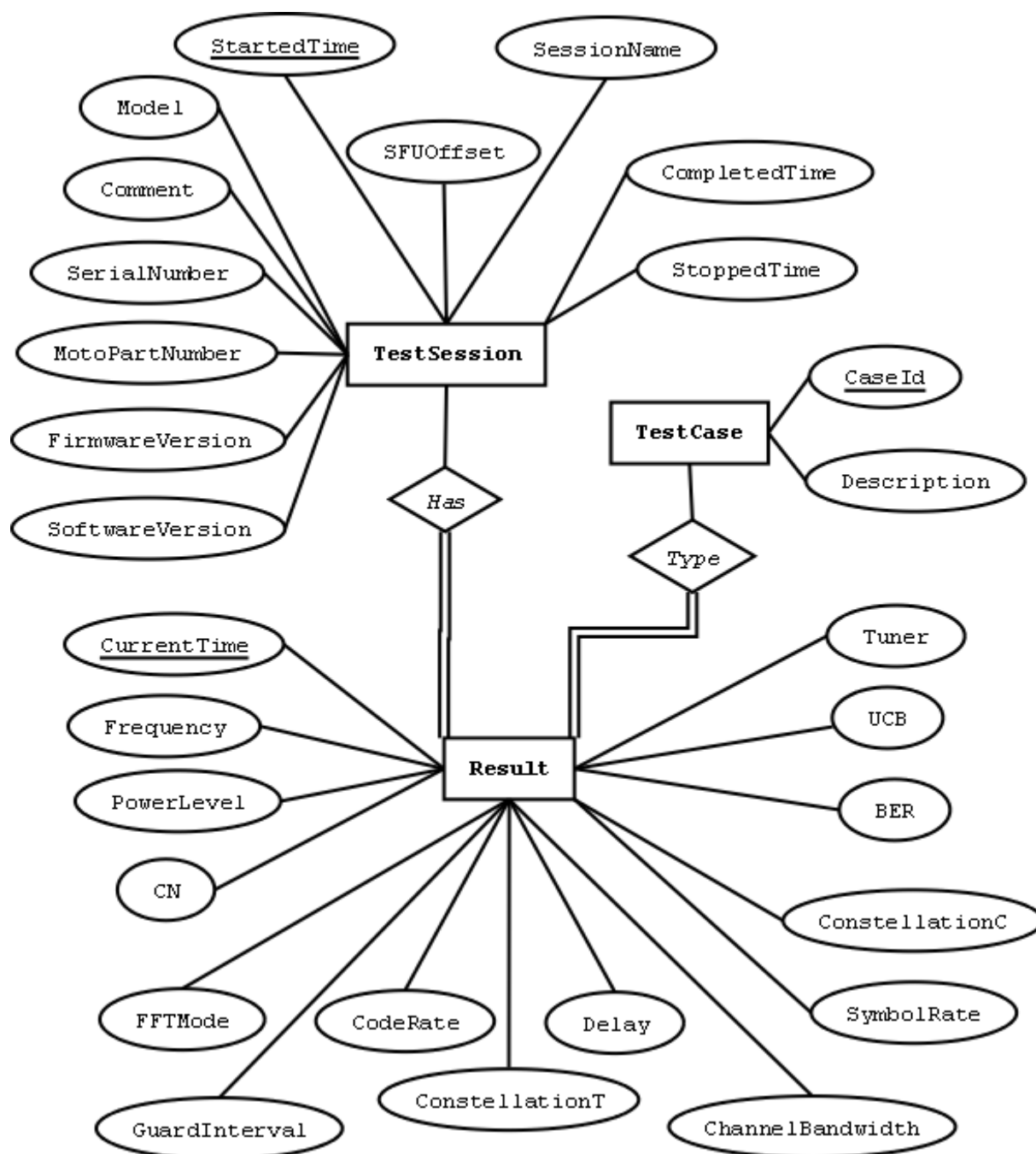


Figure 9. E-R Diagram

5.5.2. Database connection

ODBC (Open Database Connectivity) provides a standard software API method for using database management systems (DBMS). In this project, MySQL ODBC 5.1 Driver is used; meanwhile, MFC supports the CDatabase class, which brings a lot of convenience. For database connection, an OpenEx() method is chosen from the library which throws CDBException and CMemoryException. The input arguments are an ODBC connect string and a connect option. The connect string includes the driver type, server address, port number, data source, user ID and password. A connect option is selected by the user, and 0 is the default value, which means database will be opened as shared with write access, the ODBC Cursor Library DLL will not be loaded, and the ODBC connection dialog box will display only if there is not enough information to make the connection [13].

Here is an example in my program,
CDataBase m_db;
m_db.OpenEx("Driver=MySQL ODBC 5.1 Driver; Server=server.database.com; Port=3306;
Database=dvb; UID=user; PWD=password");

5.5.3. Operations on database

In this database implementation, only two operations are needed, update and insert. In the beginning of the test, test session information is inserted into TestSession table only once. At the end of the whole test, either completion time or stopped time is updated. During the test, result is inserted into Result table once an item comes out.

The syntax for insert statement in SQL is like this.

```
INSERT INTO [ table_name ] < ( [ columns_list ] ) > VALUES ( [ values_list ] )
```

If values are provided for all columns in the table in the *value_list*, column names do not have to be provided. However, if values in the *value_list* do not match the number of columns in the table or if values in the *values_list* are to be placed in specific columns in the table, column names must be explicitly stated in the *columns_list* [10].

The syntax for update statement in SQL is like this.

```
UPDATE [ table_name ] SET [ column ] = [ value ] < WHERE [ where_condition ] >
```

This statement always contain a WHERE clause, or all items in the table will be updated [10].

Moreover, there is a method called ExecuteSQL() supported by CDatabase class. It is very convenient to use, one way is to make the SQL statement in a CString format as an input argument, and the other is to transmit a pointer to a null-terminated string containing a valid SQL command [11].

The methods of the Database class are listed in the appendix.

5.6. Data Presentation

Data presentation is done in Microsoft Excel 2003 using Visual Basic for Application. The programs are written in Visual Basic, including database connection and report generator. This report tool needs to extract data from database and format it in a user friendly way.

The report tool is shown in Figure 10. There are two buttons to generate DVB-T and DVB-C report respectively. The content in the first column is the index for the test case. The second column is the session started time for single tuner or the first interface of the dual tuner. The third column is the session start time for the second interface of the dual tuner. The strings in these two columns are the conditions of the SQL query. Once the button is clicked, a report (*.xls file) is created, which retrieves data from database and arranges them in a user friendly way.

Figure 11 is the reference for different sheets. From this reference, it is quite easy for users to find out the content in each sheet.

	A	B	C
1	CaseID	S_SessStartTime	D_SessStartTime
2	T1	20100702095241	20100702095241
3	T2	20100702095241	20100702095241
4	T3	20100702095241	20100702095241
5	T4	20100702095241	20100702095241
6	T5	20100702095241	20100702095241
7	T6	20100702095241	20100702095241
8	T7	20100702095241	20100702095241
9	T8	20100702095241	20100702095241
10	T9	20100708122909	20100708122909
11			
12	C1	20100712100612	20100712100612
13	C2	20100712100612	20100712100612
14	C3	20100713144700	20100713144700
15	C4	20100713144700	20100713144700
16			
17	STB No.	VIP1963C	
18			
19			
20			
21	Generate DVBT Report		Generate DVBC Report
22			

Figure 10. The report tool

H	I	J	K	L
Reference Sheet				
S_original	Single	DVBT	Dual	D_original
	2	T1	17	
	3	T2	18	
	4	T3	19	
11	5	T4	20	26
12	6	T5	21	27
13	7	T6	22	28
14	8	T7	23	29
15	9	T8	24	30
16	10	T9	25	31
S_original	Single	DVBC	Dual	D_original
6	2	C1	10	14
7	3	C2	11	15
8	4	C3	12	16
	5	C4	13	

Figure 11. The reference sheet

6. Results

There are four ways to present data. Users can see every change of the SFU and the STB in the debug window. All results and debug information are written to a log file, and only results are put into the database. After data collection has been done, the report tool formats data automatically.

6.1. Debug window

Debug display window is used to see the result in real time. It displays all the changes both in the SFU and the STB, so it is very convenient to debug from this window.

Figure 12 is the snapshot of a running DVBTOOL.

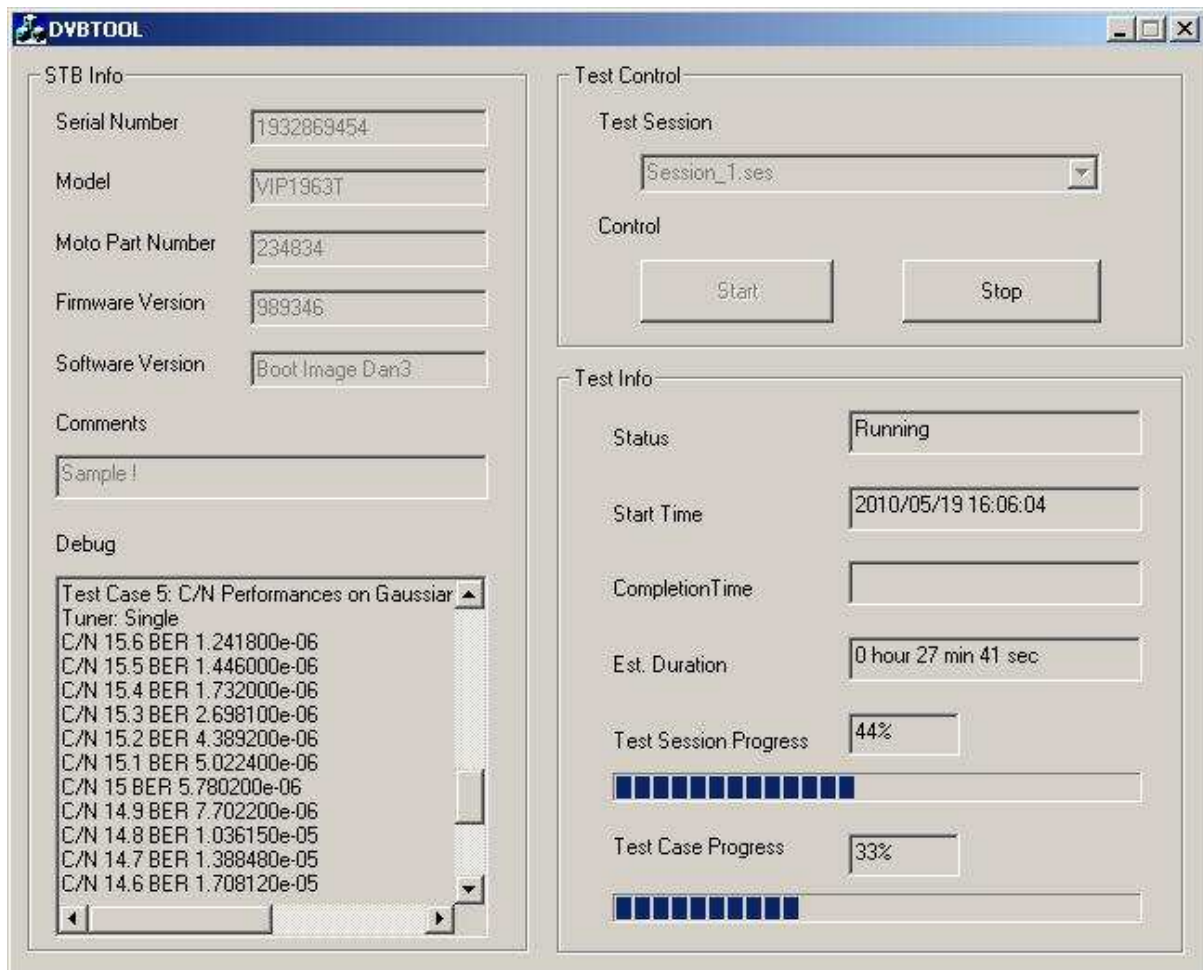
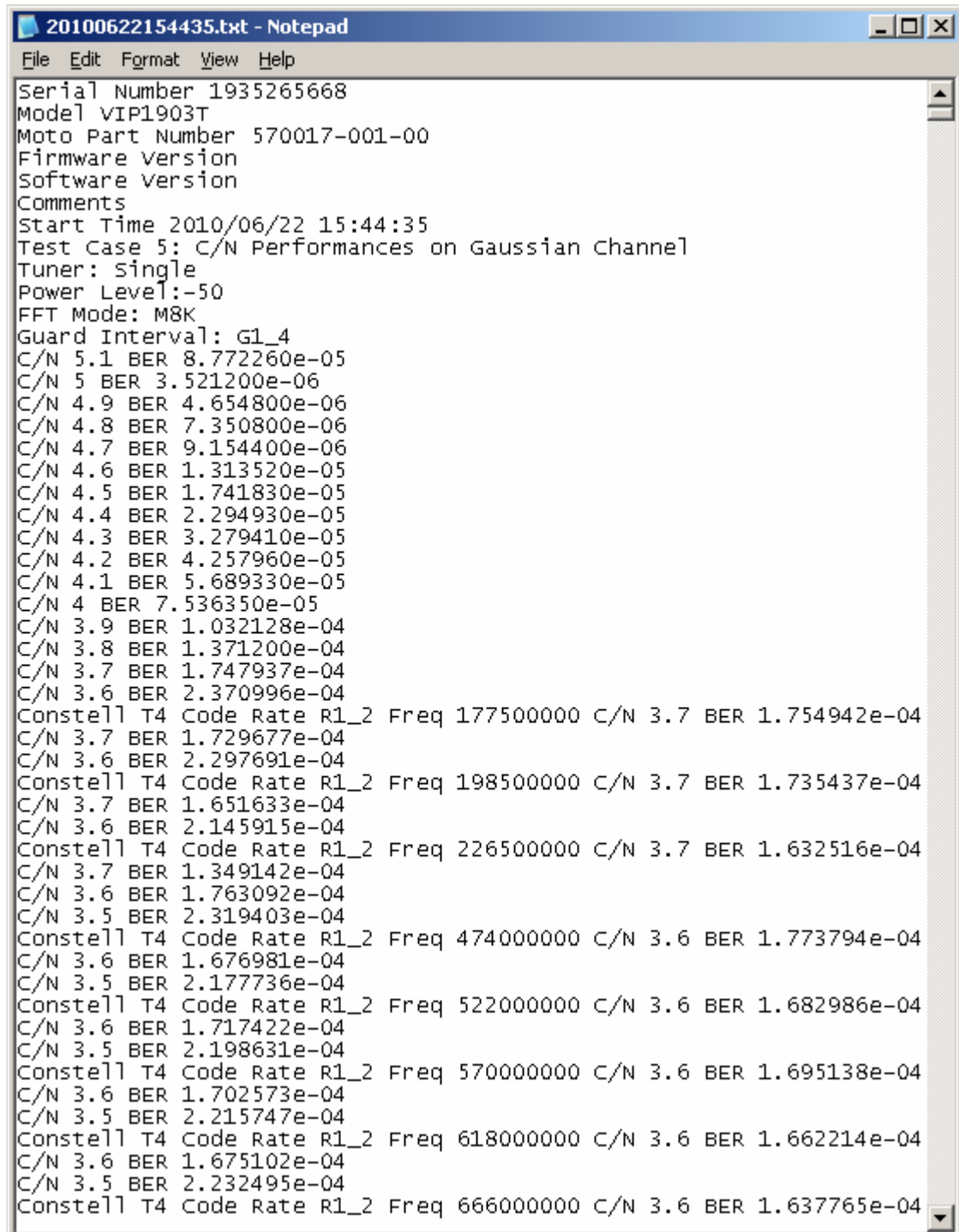


Figure 12. The running DVB measurement tool

6.2. Log file

The log file is created every time the Start button is clicked, its name is a format of system time, so it will never be duplicated. In the initialize file, the user can modify the place where to put the log and choose whether or not debug information will be written to the log.

The log file is shown in figure 13.



```
20100622154435.txt - Notepad
File Edit Format View Help
Serial Number 1935265668
Model VIP1903T
Moto Part Number 570017-001-00
Firmware Version
Software Version
Comments
Start Time 2010/06/22 15:44:35
Test Case 5: C/N Performances on Gaussian Channel
Tuner: single
Power Level:-50
FFT Mode: M8K
Guard Interval: G1_4
C/N 5.1 BER 8.772260e-05
C/N 5 BER 3.521200e-06
C/N 4.9 BER 4.654800e-06
C/N 4.8 BER 7.350800e-06
C/N 4.7 BER 9.154400e-06
C/N 4.6 BER 1.313520e-05
C/N 4.5 BER 1.741830e-05
C/N 4.4 BER 2.294930e-05
C/N 4.3 BER 3.279410e-05
C/N 4.2 BER 4.257960e-05
C/N 4.1 BER 5.689330e-05
C/N 4 BER 7.536350e-05
C/N 3.9 BER 1.032128e-04
C/N 3.8 BER 1.371200e-04
C/N 3.7 BER 1.747937e-04
C/N 3.6 BER 2.370996e-04
Constell T4 Code Rate R1_2 Freq 177500000 C/N 3.7 BER 1.754942e-04
C/N 3.7 BER 1.729677e-04
C/N 3.6 BER 2.297691e-04
Constell T4 Code Rate R1_2 Freq 198500000 C/N 3.7 BER 1.735437e-04
C/N 3.7 BER 1.651633e-04
C/N 3.6 BER 2.145915e-04
Constell T4 Code Rate R1_2 Freq 226500000 C/N 3.7 BER 1.632516e-04
C/N 3.7 BER 1.349142e-04
C/N 3.6 BER 1.763092e-04
C/N 3.5 BER 2.319403e-04
Constell T4 Code Rate R1_2 Freq 474000000 C/N 3.6 BER 1.773794e-04
C/N 3.6 BER 1.676981e-04
C/N 3.5 BER 2.177736e-04
Constell T4 Code Rate R1_2 Freq 522000000 C/N 3.6 BER 1.682986e-04
C/N 3.6 BER 1.717422e-04
C/N 3.5 BER 2.198631e-04
Constell T4 Code Rate R1_2 Freq 570000000 C/N 3.6 BER 1.695138e-04
C/N 3.6 BER 1.702573e-04
C/N 3.5 BER 2.215747e-04
Constell T4 Code Rate R1_2 Freq 618000000 C/N 3.6 BER 1.662214e-04
C/N 3.6 BER 1.675102e-04
C/N 3.5 BER 2.232495e-04
Constell T4 Code Rate R1_2 Freq 666000000 C/N 3.6 BER 1.637765e-04
```

Figure 10. The log file

6.3. Database

Database is optional in this project, the user can choose to use it or not. An argument (DB_USE) is used to control the database connectivity in the initialize file (*.ini file). If this value equals 0, database is not used, and if it equals to 1, the database is used.

The contents of the Result table in the database are shown in figure 14.

CurrentTime	Frequency	Power	CN	Cons	Channe	FFTI	Guarc	Codef	Del	Constell	Sym	Tuner	UCB	BER	SessionStartedTime	TestCaseId
20100702100114	177500000	-50	3.6	T4	BW_7	M8K	G1_4	R1_2	0	NULL	0	0	0	0.0001625183	20100702095241	T5
20100702100234	198500000	-50	3.6	T4	BW_7	M8K	G1_4	R1_2	0	NULL	0	0	0	0.0001678758	20100702095241	T5
20100702100354	226500000	-50	3.6	T4	BW_7	M8K	G1_4	R1_2	0	NULL	0	0	0	0.0001610375	20100702095241	T5
20100702100512	474000000	-50	3.6	T4	BW_8	M8K	G1_4	R1_2	0	NULL	0	0	0	0.0001575163	20100702095241	T5
20100702100632	522000000	-50	3.6	T4	BW_8	M8K	G1_4	R1_2	0	NULL	0	0	0	0.0001637357	20100702095241	T5
20100702100752	570000000	-50	3.6	T4	BW_8	M8K	G1_4	R1_2	0	NULL	0	0	0	0.000161493	20100702095241	T5
20100702100911	618000000	-50	3.6	T4	BW_8	M8K	G1_4	R1_2	0	NULL	0	0	0	0.0001599938	20100702095241	T5
20100702101031	666000000	-50	3.6	T4	BW_8	M8K	G1_4	R1_2	0	NULL	0	0	0	0.0001630902	20100702095241	T5
20100702101150	714000000	-50	3.6	T4	BW_8	M8K	G1_4	R1_2	0	NULL	0	0	0	0.0001738419	20100702095241	T5
20100702101310	762000000	-50	3.6	T4	BW_8	M8K	G1_4	R1_2	0	NULL	0	0	0	0.0001663888	20100702095241	T5
20100702101430	810000000	-50	3.6	T4	BW_8	M8K	G1_4	R1_2	0	NULL	0	0	0	0.000164232	20100702095241	T5
20100702101548	858000000	-50	3.6	T4	BW_8	M8K	G1_4	R1_2	0	NULL	0	0	0	0.0001534987	20100702095241	T5
20100702102422	177500000	-50	5.3	T4	BW_7	M8K	G1_4	R2_3	0	NULL	0	0	0	0.0001541993	20100702095241	T5
20100702102541	198500000	-50	5.3	T4	BW_7	M8K	G1_4	R2_3	0	NULL	0	0	0	0.0001595281	20100702095241	T5
20100702102701	226500000	-50	5.3	T4	BW_7	M8K	G1_4	R2_3	0	NULL	0	0	0	0.000156779	20100702095241	T5
20100702102820	474000000	-50	5.3	T4	BW_8	M8K	G1_4	R2_3	0	NULL	0	0	0	0.0001498529	20100702095241	T5
20100702102940	522000000	-50	5.3	T4	BW_8	M8K	G1_4	R2_3	0	NULL	0	0	0	0.0001574244	20100702095241	T5
20100702103059	570000000	-50	5.3	T4	BW_8	M8K	G1_4	R2_3	0	NULL	0	0	0	0.0001538357	20100702095241	T5
20100702103219	618000000	-50	5.3	T4	BW_8	M8K	G1_4	R2_3	0	NULL	0	0	0	0.0001532393	20100702095241	T5
20100702103339	666000000	-50	5.3	T4	BW_8	M8K	G1_4	R2_3	0	NULL	0	0	0	0.0001595812	20100702095241	T5
20100702103457	714000000	-50	5.3	T4	BW_8	M8K	G1_4	R2_3	0	NULL	0	0	0	0.0001680718	20100702095241	T5
20100702103617	762000000	-50	5.3	T4	BW_8	M8K	G1_4	R2_3	0	NULL	0	0	0	0.0001599775	20100702095241	T5
20100702103737	810000000	-50	5.3	T4	BW_8	M8K	G1_4	R2_3	0	NULL	0	0	0	0.0001527716	20100702095241	T5
20100702103859	858000000	-50	5.3	T4	BW_8	M8K	G1_4	R2_3	0	NULL	0	0	0	0.0001513112	20100702095241	T5
20100702104805	177500000	-50	6.2	T4	BW_7	M8K	G1_4	R3_4	0	NULL	0	0	0	0.0001794035	20100702095241	T5
20100702104924	198500000	-50	6.2	T4	BW_7	M8K	G1_4	R3_4	0	NULL	0	0	0	0.0001812336	20100702095241	T5
20100702105044	226500000	-50	6.2	T4	BW_7	M8K	G1_4	R3_4	0	NULL	0	0	0	0.0001793464	20100702095241	T5
20100702105204	474000000	-50	6.2	T4	BW_8	M8K	G1_4	R3_4	0	NULL	0	0	0	0.0001749816	20100702095241	T5
20100702105323	522000000	-50	6.2	T4	BW_8	M8K	G1_4	R3_4	0	NULL	0	0	0	0.0001784844	20100702095241	T5
20100702105443	570000000	-50	6.2	T4	BW_8	M8K	G1_4	R3_4	0	NULL	0	0	0	0.0001782209	20100702095241	T5
20100702105602	618000000	-50	6.2	T4	BW_8	M8K	G1_4	R3_4	0	NULL	0	0	0	0.0001756147	20100702095241	T5
20100702105722	666000000	-50	6.2	T4	BW_8	M8K	G1_4	R3_4	0	NULL	0	0	0	0.0001821854	20100702095241	T5
20100702105841	714000000	-50	6.2	T4	BW_8	M8K	G1_4	R3_4	0	NULL	0	0	0	0.0001887152	20100702095241	T5

Figure 14. The contents in the Result table

6.4. Report in Excel

The report tool extracts data from database by SQL query and formats them in a very nice way. The two pictures below show the differences quite clearly.

Figure 15 is an example that shows the data directly retrieved from database.

	A	B	C	D
1	ConstellationT	CodeRate	Frequency	CN
2	T4	R1_2	198500000	5.7
3	T4	R1_2	666000000	5.8
4	T4	R2_3	198500000	9.7
5	T4	R2_3	666000000	9
6	T4	R3_4	198500000	12.7
7	T4	R3_4	666000000	11.8
8	T16	R1_2	198500000	11.2
9	T16	R1_2	666000000	11
10	T16	R2_3	198500000	14.9
11	T16	R2_3	666000000	14.5
12	T16	R3_4	198500000	18.2
13	T16	R3_4	666000000	17.5
14	T64	R1_2	198500000	15.9
15	T64	R1_2	666000000	15.6
16	T64	R2_3	198500000	20
17	T64	R2_3	666000000	19.7
18	T64	R3_4	198500000	23.1
19	T64	R3_4	666000000	22.5

Figure 15. The data in Excel before formatted

Figure 16 is the result after data is formatted.

	A	B	C	D	E
1	T6	C/N on 0dB Echo Channel			
2					
3	Power	FFT Mode	Guard Interval	Delay	
4	-50	M8K	G1_4	1.95E-06	
5					
6					
7				CN	
8				7MHz-VHF	8MHz-UHF
9	Constellation	Code Rate	Specification	198500000	666000000
10	T4	R1_2	8.8	5.7	5.8
11	T4	R2_3	13.7	9.7	9
12	T4	R3_4	17.4	12.7	11.8
13	T16	R1_2	13.3	11.2	11
14	T16	R2_3	17.9	14.9	14.5
15	T16	R3_4	22.1	18.2	17.5
16	T64	R1_2	19	15.9	15.6
17	T64	R2_3	23.2	20	19.7
18	T64	R3_4	27.6	23.1	22.5

Figure 116. The data in Excel after formatted

7. Conclusion

In this project, the DVB measurement tool connects the SFU and the STB by socket communication. In the beginning it loads the test session, then sets arguments in the SFU by remote control commands. The output signal from the SFU is sent to the STB through an RF cable. The software in the STB measures the quality of the incoming signal and the DVB tool records the stable data. During this period, the application displays debug information in real time and saves the results both in the log file (*.txt file) and the database. When data collection is done, the report tool extracts and formats data automatically. The whole system has resulted in a fully working and very useful tool.

With this automatic DVB performance measurement tool, it becomes quite convenient to repeat the measurements for different products. Under control of the computer, the output signal from the SFU is more accurate, and no manual labor is wasted after the Start button has been clicked. So this tool increases the measurement quality and decreases the development time a lot.

Besides, all these test cases are configurable simply by modifying the test session (*.ses file). The user can run a full test session or some of the test cases automatically with different modes for a large range of frequencies. But when testing manually in practice, one selects a limited number of modes and frequencies due to the work load. A typical problem is that the performance is uneven over the whole frequency range. With an automatic tool, one can test more frequencies than manually to find out problems.

Further on, this system provides an easy way to make comparison between the performances of the STBs. Once there are results stored in the database, the report tool can generate reports for the products of any type. Especially during the development phase of a product with optimal performance, software and hardware is often updated. The report tool helps to find the performance difference after any change much faster than if done manually.

Finally, the system can also be used to collect statistical data for a product model. Test several product individuals and make sure the results stored in the database. After all tests complete, one can easily obtain the minimum value, the maximum value, the mean value and standard deviation from the database for any given parameters.

8. Further development

Currently only the test cases derived from NorDig Unified 2.1 have been implemented, however, the system can easily be extended to other standards.

In DVB-T and DVB-C environment, it always takes some time for the STB to become stable, but this period changes if different fields in the environment settings have changed. To make this DVB measurement tool suitable for a high range of products, longer time (26 seconds) is selected in the project. However, the period could be shorter according to specific settings which would decrease the testing time a lot. Also it would be possible to write an algorithm to determine suitable time for any mode of operation.

In this project, the STB should be tuned to different frequencies many times, before the next turning procedure, a stop command is sent to the STB, but the STB needs time to react. If it is not in a real stop state, the next tuning procedure will fail, and the whole test system will hang in the middle. To deal with this, when the stop command is sent, the system will wait two seconds for the STB to react. But if it is not long enough, the whole test system hangs after all. A solution is to improve the software in the STB, which can provide more robust API. Once the stop command is sent, a return value could be feed back. According to this value, it is quite obvious to see whether the STB has stopped or not.

Although users can see the debug information from both the debug window and the log file, it is better to have a debug class separately instead of embedded in the CWorkerThread. This class could be accessed by all classes in the project, and then more information would be displayed, which helps to find error much more quickly.

Further on, if a problem occurs in the STB, DVBT00L could not recover. One could add problem detection and reboot function to resume the test.

9. References

- [1] http://homepages.uel.ac.uk/u9703493/S03AG_TCP.gif (retrieved: 2010-07-01)
- [2] http://www.motorola.com/staticfiles/Business/Products/TV%20Video%20Distribution/Set-tops/IP%20Set-tops/VIP1963T/IMAGES/_Staticfiles/VIP1963T_MD.jpg (retrieved: 2010-07-01)
- [3] <http://www2.rohde-schwarz.com/live/rs/mediadb/pspic/image/24/import45b08bd5c2254.jpg> (retrieved: 2010-07-01)
- [4] Operating manual R&S broadcast test system
- [5] http://upload.wikimedia.org/wikipedia/commons/thumb/8/8f/QPSK_Gray_Coded.svg/200px-QPSK_Gray_Coded.svg.png (retrieved: 2010-07-01)
- [6] http://upload.wikimedia.org/wikipedia/commons/thumb/1/1e/16QAM_Gray_Coded.svg/200px-16QAM_Gray_Coded.svg.png (retrieved: 2010-07-01)
- [7] http://www.nordig.org/pdf/NorDig-Unified_ver_2.1.pdf (retrieved: 2010-07-01)
- [8] Douglas E. Comer. 2005. Internetworking with TCP/IP, Vol 1: Principles, Protocols, and Architectures. Fifth Edition. Pearson Prentice Hall
- [9] John Proakis and Masoud Salehi. Digital Communications. Fifth Edition. Mcgraw-Hill
- [10] HectorGarcia-Molina, Jeffrey D. Ullman, Jennifer Widom. Database Systems: The Complete Book: International Version. Second Edition. Pearson Higher Education.
- [11] Roger E. Sanders. 1998. ODBC 3.5 Developer's Guide. Mcgraw-Hill.
- [12] Bjarne Stroustrup. 2000. The C++ Programming Language: Special Edition. Addison-Wesley.
- [13] MSDN library, Microsoft Corporation
- [14] Jeff Prosise. 1999. Programming Windows with MFC. Second Edition. Microsoft Press.

Appendix

Here are the .h files used in this project, including SFU.h, STB.h, Database.h, CWorkerThread.h and CDVBTOOLDlg.h.

SFU.h file

```
#ifndef _SFU_H_
#define _SFU_H_

#include<Winsock2.h>
#include<string>
#include<iostream>
#include<vector>
#include<stdlib.h>
using namespace std;

class SFU{

private:

    //connection parameters
    SOCKET PCsocket;
    SOCKADDR_IN SFUaddr;
    WSADATA wsaData;

    // common attributes
    string frequency;
    int freq;
    string powlevel;
    double pow;
    string powoffset;
    double offs;
    string standard;
    string noise;
    string noiseCN;
    double noisCN;

    //DVBC attributes
    string symbolrate;
    int symrate;
    string constellationC;

    //DVBT attributes
    string channelbandwidth;
    int channelband;
    string fftmode;
    string guardinterval;
```

```

string coderate;
string constellationT;
string del;
double delay;
string state;

//receive data from SFU, frequency, pow, offset...etc
string RecData();

//send data from computer to set SFU parameters
void SendData(string da);

public:

//DVBC constructor
SFU(string,string,string,string,string,string,string,string);

//DVBT constructor
SFU(string,string,string,string,string,string,string, string,string,string);

//Default constructor DVBT mode
SFU();

void Stop();

//Initialize, build connection
void SFUini(string s);

//cast functions
const char* StringToChar(string ss);
string DoubleToString(double dou);
string IntToString(int i);
double StringToDouble(string ss);
int StringToInt(string ss);

// common functions
void SetFrequency(string f);
void SetFrequencyVal(int fv);
string GetFrequency();
int GetFrequencyVal();

void SetPowLevel(string p);
void SetPowLevelVal(double pv);
string GetPowLevel();
double GetPowLevelVal();

void SetPowOffset(string po);
void SetPowOffsetVal(double pov);
string GetPowOffset();
double GetPowOffsetVal();

```



```

void SetStandard(string s);
string GetStandard();

void SetNoise(string n);
string GetNoise();

void SetNoiseCN(string ncn);
void SetNoiseCNVal(double ncnv);
string GetNoiseCN();
double GetNoiseCNVal();

//DVBC function
void SetSymbolRate(string sr);
void SetSymbolRateVal(int srv);
string GetSymbolRate();
int GetSymbolRateVal();

void SetConstellationDVBC(string cc);
string GetConstellationDVBC();

//DVBT functions
void SetChannelBand(string cb);
void SetChannelBandVal(int cbv);
string GetChannelBand();
int GetChannelBandVal();

void SetGuardInterval(string gi);
string GetGuardInterval();

void SetFftMode(string fm);
string GetFftMode();

void SetCodeRate(string cr);
string GetCodeRate();

void SetConstellationDVBT(string ct);
string GetConstellationDVBT();

void SetDelay(string s);
string GetDelay();

void SetState(string s);
string GetState();

};

#endif

```

STB.h file

```
#ifndef _STB_H_
#define _STB_H_

#include "stdafx.h"
#include<stdio.h>
#include<Winsock2.h>
#include<string>
#include<iostream>
#include<vector>
#include<math.h>
#include <stdlib.h>
using namespace std;
#pragma comment(lib, "WS2_32.LIB")

class STB
{
private:

    double BERVal;
    double SNVal;
    int UCBVal;

    string quality;
    string BER;
    string SN;
    string UCB;

    double stableBERVal;
    double stableSNVal;
    int stableUCBVal;

    string stablequality;
    string stableBER;
    string stableSN;
    string stableUCB;

    vector<string> qualities;
    int UCBSum;

    SOCKET controlsocket;
    SOCKADDR_IN STBaddr;
    WSADATA wsaData;
    int result;

public:

    STB();
```

```
~STB());

void STBIni(string s);
void TuneDVBT(string f,string i,string b);
void TuneDVBC(string f,string i,string m,string s);
void Stop();
void RealStop();

string GetInstantQuality();
string GetStableQuality(int second);

string GetBER();
string GetSN();
string GetUCB();

double GetBERVal();
double GetSNVal();
int GetUCBVal();

string GetStableBER();
string GetStableSN();
string GetStableUCB();

double GetStableBERVal();
double GetStableSNVal();
int GetStableUCBVal();

int GetStableUCBSum();

double StringToDouble(string ss);
int StringToInt(string ss);

void reboot();
};

#endif
```

Database.h file

```
#ifndef _DATABASE_H_  
#define _DATABASE_H_
```

```
#include "stdafx.h"  
#include "sqlext.h"  
#include<string>  
#include<Winsock2.h>  
#include<string>  
#include<iostream>  
#include<vector>  
#include<stdlib.h>  
using namespace std;
```

```
class DataBase
```

```
{  
private:
```

```
    CDatabase m_db;
```

```
public:
```

```
    //Test Session Attributes  
    string StartTime;  
    string CompleteTime;  
    string StopTime;  
    string SFUOffs;  
    string SessName;  
    string Mod;  
    string Comm;  
    string SerialNo;  
    string MotoPartNo;  
    string FirmwareV;  
    string SoftwareV;
```

```
    //Result Attributes  
    string CurrTime;  
    string SessStartedTime;  
    string TCaseId;  
    int Freque;  
    double PowerLev;  
    double CtoN;
```

```
    string ConstellT;  
    string ChannelBand;  
    string FFTM;  
    string GuardInterv;  
    string CodingRate;  
    double Del;
```

```
string ConstellC;  
double SymRate;  
  
int TunerNo;  
int UCBsum;  
double BERval;  
  
DataBase();  
bool DataBaseIni();  
bool UpdateTestSession();  
bool InsertTestSession();  
bool InsertDVBCResult();  
bool InsertDVBTRResult();  
void Close();  
};  
  
#endif
```

CWorkerThread.h file

```
#if !defined(AFX_WORKERTHREAD_H__90395F70_9AE7_4D8E_9395_BCC513FAD4D7__INCLUDED_)
#define
AFX_WORKERTHREAD_H__90395F70_9AE7_4D8E_9395_BCC513FAD4D7__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// WorkerThread.h : header file
//
#include "STB.h";
#include "SFU.h"
#include "DataBase.h"
#include <string>
#include <vector>
#include<map>
#include<sstream>
#include<fstream>
#include<iostream>
using namespace std;

#define WM_INI WM_USER+1
#define WM_STOPPING WM_USER+2

////////////////////////////////////
// CWorkerThread thread

class CWorkerThread : public CWinThread
{
    DECLARE_DYNCREATE(CWorkerThread)
protected:
    CWorkerThread();    // protected constructor used by dynamic creation

// Attributes
public:

    SFU sfu;
    STB stb;
    DataBase db;

    char dvbtype;

    int est1;
    int est2;
    int est3;
    int est4;
    long est5;
```

```

long est6;
long est7;
long est8;
int est9;

long cest1;
long cest2;
int cest3;
int cest4;

int testcases[9];
int casesize[9];
int tuner[9];

int ctestcases[4];
int ccasesize[4];
int ctuner[4];

// DVBT test cases

// Test case 1 parameters, center frequencies
vector<string> vec_freq1_vhf;
vector<string> vec_freq1_uhf;
string static_power1;
string static_FFTmode1;
string static_guardinterval1;
string static_constell1;
string static_coderate1;

//Test case 2 paremeters, frequency offset
vector<string> vec_freq2_vhf;
vector<string> vec_freq2_uhf;
string static_freqoffset;
string static_power2;
string static_FFTmode2;
string static_guardinterval2;
string static_constell2;
string static_coderate2;

//Test case 3 paremeters, frequency offset
vector<string> vec_freq3_vhf;
vector<string> vec_freq3_uhf;
string static_power3;
string static_FFTmode3;
string static_guardinterval3;
string static_constell3;
string static_coderate3;

//Test case 4 parameters, modes

```

```

string static_freq4_uhf;
string static_power4;
vector<string> vec_FFTmode4;
vector<string> vec_constell4;
vector<string> vec_guardinterval4;
vector<string> vec_coderate4;

//Test case 5 parameters, C/N performance on Gaussian Channel
string static_power5;
string static_FFTmode5;
string static_guardinterval5;
vector<string> vec_constell5;
vector<string> vec_coderate5;
vector<string> vec_freq5_vhf;
vector<string> vec_freq5_uhf;
double std5_all[15];
map<string,int> map_std5;

//Test case 6 parameters, C/N performance on 0 dB channel
string static_power6;
string static_FFTmode6;
string del6;
double delay6;
string static_guardinterval6;
vector<string> vec_constell6;
vector<string> vec_coderate6;
string static_freq6_vhf;
string static_freq6_uhf;
double std6_all[9];
map<string,int> map_std6;

//Test case 7 parameters, minimum input on gaussian channel
string static_FFTmode7;
string static_guardinterval7;
vector<string> vec_constell7;
vector<string> vec_coderate7;
vector<string> vec_freq7_vhf;
vector<string> vec_freq7_uhf;
double std7_vhf[15];
double std7_uhf[15];
map<string,int> map_std7_vhf;
map<string,int> map_std7_uhf;

//Test case 8 parameters, minimum input on 0 dB channel
string static_FFTmode8;
string static_guardinterval8;
vector<string> vec_constell8;
vector<string> vec_coderate8;
vector<string> vec_del8;
string static_freq8_vhf;

```



```

string static_freq8_uhf;
double std8_vhf[9];
double std8_uhf[9];
map<string,int> map_std8_vhf;
map<string,int> map_std8_uhf;

//Test case 9 parameters, maximum receiver signal input level
string static_FFTmode9;
string static_constell9;
string static_freq9_uhf;
vector<string> vec_pow9;

// DVBC test cases

//DVBC testcase 1
vector<string> vec_cfreq1;
vector<string> vec_cconstell1;
string static_symrate1;
string static_cpow1;
double cstd1[4];
map<string,int> map_cstd1;

//DVBC testcase 2
vector<string> vec_cfreq2;
vector<string> vec_cconstell2;
string static_symrate2;
double cstd2[4];
map<string,int> map_cstd2;
double cpow2[4];
map<string,int> map_cpow2;

//DVBC testcase 3
vector<string> vec_cconstell3;
string static_cfreq3;
vector<string> vec_symrate3;
double static_cmaxpow3;
double cminpow3[5];
map<string,int> map_cminpow3;
double ccn3[5];
map<string,int> map_ccn3;

//DVBC testcase4
string static_cfreq4;
string static_cpow4;
vector<string> vec_symrate4;
vector<string> vec_cconstell4;
double ccn4[5];
map<string,int> map_ccn4;

```

```

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CWorkerThread)
    public:
    virtual BOOL InitInstance();
    virtual int ExitInstance();
    //}}AFX_VIRTUAL

// Implementation
protected:
    virtual ~CWorkerThread();
    // Generated message map functions
    //{{AFX_MSG(CWorkerThread)
        // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG
afx_msg LONG OnInitialize(WPARAM wParam, LONG lParam);
afx_msg LONG OnStopping(UINT wParam, LONG lParam);

void ReadSessionFile();
void DebugDisplay(string s);

// data conversions
double StringToDouble(string ss);
int StringToInt(string ss);
string IntToString(int i);
string DoubleToString(double dou);

//DVBT functions
bool ExeTestCase1(string s);
bool ExeTestCase2(string s);
bool ExeTestCase3(string s);
bool ExeTestCase4(string s);

bool ExeTestCase5(string s);
void Case5Initialize();

bool ExeTestCase6(string s);
void Case6Initialize();

bool ExeTestCase7(string s);
void Case7Initialize();

bool ExeTestCase8(string s);
void Case8Initialize();

```

```

bool ExeTestCase9(string s);

// DVBC functions
bool ExeDVBCTestCase1(string s);
void CCase1Initialize();

bool ExeDVBCTestCase2(string s);
void CCase2Initialize();

bool ExeDVBCTestCase3(string s);
void CCase3Initialize();

bool ExeDVBCTestCase4(string s);
void CCase4Initialize();

DECLARE_MESSAGE_MAP()
};

////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous
line.

#endif
// !defined(AFX_WORKERTHREAD_H__90395F70_9AE7_4D8E_9395_BCC513FAD4D7
__INCLUDED_)

```

CDVBTOOLDlg.h file

```
// DVBTOOLDlg.h : header file
```

```
#if !defined(AFX_DVBTOOLDLG_H__6DB23458_C805_4F29_B6BA_BA1CFF825D4B__INCLUDED_)
```

```
#define
```

```
AFX_DVBTOOLDLG_H__6DB23458_C805_4F29_B6BA_BA1CFF825D4B__INCLUDE  
D_
```

```
#if _MSC_VER > 1000
```

```
#pragma once
```

```
#endif // _MSC_VER > 1000
```

```
////////////////////////////////////
```

```
// CDVBTOOLDlg dialog
```

```
#include "WorkerThread.h"
```

```
#define WM_DISPLAY WM_USER+3
```

```
#define WM_STOPPED WM_USER+4
```

```
#define WM_SESS_PROC WM_USER+5
```

```
#define WM_CASE_PROC WM_USER+6
```

```
#define WM_EST_TIME WM_USER+7
```

```
class CDVBTOOLDlg : public CDialog
```

```
{
```

```
// Construction
```

```
public:
```

```
    CDVBTOOLDlg(CWnd* pParent = NULL);    // standard constructor
```

```
    CWorkerThread* m_pWorkerThread;
```

```
// Dialog Data
```

```
//{{AFX_DATA(CDVBTOOLDlg)
```

```
enum { IDD = IDD_DVBTOOL_DIALOG };
```

```
CEdit m_est;
```

```
CProgressCtrl m_bar_case_proc;
```

```
CEdit m_case_proc;
```

```
CEdit m_sess_proc;
```

```
CProgressCtrl m_bar_sess_proc;
```

```
CEdit m_debug;
```

```
CEdit m_sn;
```

```
CEdit m_model;
```

```
CEdit m_mpn;
```

```
CEdit m_sv;
```

```
CEdit m_fv;
```

```
CEdit m_completiontime;
```

```
CButton    m_stop;
```

```
CButton    m_start;
```

```

    CEdit m_starttime;
    CEdit m_status;
    CEdit m_comments;
    CComboBox m_combo_testsession;
    CString m_getstring;
//}}AFX_DATA

// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CDVBTOOLDlg)
protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
//}}AFX_VIRTUAL

// Implementation
protected:
    HICON m_hIcon;
    void LoadIniFile();
    void ListSessionFiles();
    string IntToString(int i);

    // Generated message map functions
//{{AFX_MSG(CDVBTOOLDlg)
    virtual BOOL OnInitDialog();
    afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
    afx_msg void OnPaint();
    afx_msg HCURSOR OnQueryDragIcon();
    afx_msg void OnButtonStart();
    afx_msg void OnButtonPause();
    afx_msg void OnButtonStop();
    afx_msg void OnSelchangeComboTestsession();
//}}AFX_MSG

afx_msg LRESULT OnDisplay(UINT wParam, LONG lParam);
afx_msg LRESULT OnStopped(WPARAM wParam, LONG lParam);
afx_msg LRESULT OnSessionProcess(WPARAM wParam, LONG lParam);
afx_msg LRESULT OnCaseProcess(WPARAM wParam, LONG lParam);
afx_msg LRESULT OnEstTime(WPARAM wParam, LONG lParam);

DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous
// line.

#endif
// !defined(AFX_DVBTOOLDLG_H__6DB23458_C805_4F29_B6BA_BA1CFF825D4B__I
NCLUDED_)

```