Table of Contents

Chapter 3

Part II

Part III

List of Figures and Tables

Preface

Ophthalmologic Sonography aids in the insertion of prosthetic lenses by allowing accurate measurement of the eyes and in diagnosis and tracking of orbital cavity tumors, separated retinas (retinal detachments), and other ailments of the eye and its surrounding tissue. High frequency transducers of 5-15 MHz, are made exclusively to study the eyes.

A-scan Biometry systems are Ultrasound based dimension measurement devices and i a basic item of any ophthalmology clinic or hospital. As biomedical engineers, we emphasized on enhancing the local medical devices' industry by *proposing* new designs and *developing* the technological algorithms to overcome all the problems and drawbacks of currently used A-scan Biometry systems, to cope with the rapid rate of development in worldwide medical devices industry.

The intended use of the A-scan biometry system is to measure the axial length, corneal radii, anterior chamber length (ACD) and the calculation of the IOL power required for the preoperative lens implant.

In order to motivate the local market and adapt to changes in the global industry, we will follow a progressive approach to reach our prospective goal. We hope our project acts as the corner stone of A-scan Biometry systems industry here in Egypt which supported by its simplified concept and component requirements abides well to the current National Economic and Technological conditions.

Part I

Ground Work

Chapter

1

Introduction

1.1 Introduction

Sonography is now firmly established as one of the most useful diagnostic tests because it provides reliable structural, functional, and hemodynamic information about the different systems of the human body.

Ophthalmic Sonogrpahy continue to evolve, and the development of new modalities improves its diagnostic capability. The A-Scan examination is continuously being refined through better display of ocular anatomy, more reliable quantitation, and more precise assessment of ocular health. The clinical application of this versatile diagnostic technique is performed and enhanced by e-DIOLA scan (Enhanced Diagnostic Intraocular Lens Power A-scan).

We'll start with introducing our project's vision and scope with the statement of need followed by a statistical analysis as rationalization. In order to fully comprehend the mission of e-DIOLA scan the reader will be provided with some brief grounding knowledge about the eye's biology and the physics of A-scan systems. At the end of this chapter we'll define the sequence followed throughout this document in order to keep the reader on track of our system design.

1.2 Statement of need

Performing eye dimension measurement and intraocular lens (IOL) power calculation is the initial step to be done in any Cataract surgery Ophthalmology clinic or hospital in Egypt or worldwide. Our basic concept simply depends on the ultimately safe Ultrasonic waves to identify thicknesses and depths of various tissues in the eye and hence performing further lens power calculations.

Recent Studies have shown a remarkable prevalence of ophthalmic diseases. According to the World Health Organization (WHO), cataracts are the leading cause of blindness in the world. This eye disease affects about one in every six people. Furthermore, there are 400,000 new cases each year.

Cataract surgery involves substitution of natural lens by an artificial (IOL) characterized by the exact refractive power as the natural one. Lens power is a function of eye dimensions and curvature values. Accuracy level is crucial as an error of 0.4 mm in dimensions measurement would result in one Diopter error in power calculation.

Our project will cover all aspects of a device development cycle including : Building our knowledge base, surveying competitive systems in the market, identifying specific needs of system users in Egypt and seeking continuous feedback throughout the implementation of the system and providing the full documentation user and service-based to ensure upgrading and efficient utilization of system features and tools.

We implemented a low cost, highly featured, well accommodated to Egypt Ophthalmologists' requirements; enhanced A-scan Biometry system, providing a fully designed, implemented, tested and documented prototype of the device. A software package including a friendly user interface, setup files, help files and a softcopy of user & service technical manuals with troubleshooting section, available to act as the initial step in the manufacturing process by a National Medical

Devices Company and eventually substituting the need to import such systems from other countries.

1.3 Biology of the eye

The human eye is a complex anatomical device that remarkably demonstrates the architectural wonders of the human body. It is about 0.9 inches (24 mm) in diameter and is not perfectly round, being slightly flattened in the front and back.

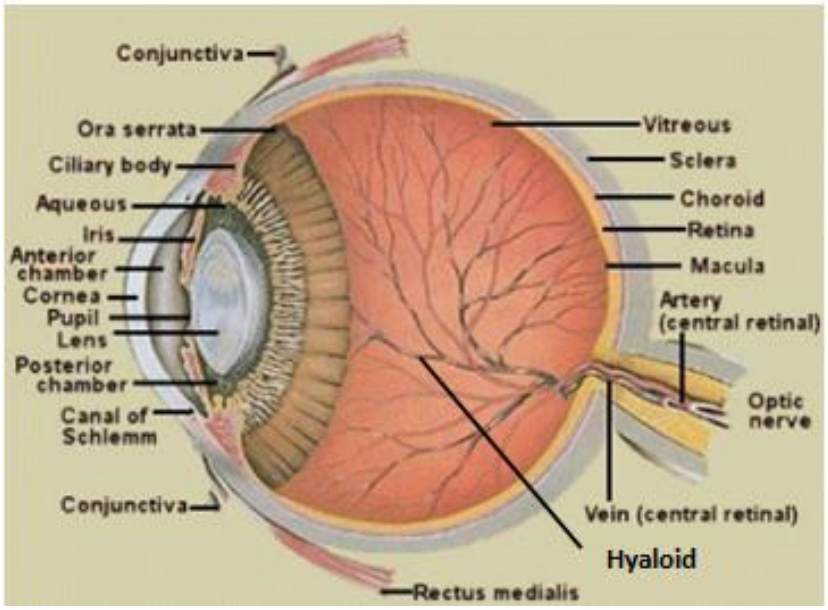


Figure 1.3-1 Anatomy of the eye

The eye consists of three layers:

1. The outer fibrous or sclera
2. The middle uveal or choroid layer
3. The inner nervous layer or retina

Internally the eye is divided into two cavities:

1. The anterior cavity filled with the watery aqueous fluid,
2. The posterior cavity filled with gel-like vitreous fluid.

1st section: The first layer, which is the outer fibrous layer, encasing and protecting the eyeball consists of two parts—the cornea and the sclera:

- **Cornea:** it is a strong clear bulge located at the front of the eye, which bends incoming light onto the lens inside the eye. The cornea is a thin membrane which has an index of refraction of approximately (n = 1.38). The cornea contributes to the image-forming process by refracting light entering the eye. It covers the iris, pupil, and anterior chamber.

- **Conjunctiva:** A fine mucus membrane that covers the cornea, and also lines the eyelid.

- **Sclera:** It is a dense, tough, opaque coat that protects the outside of the eye-ball. This is the part of the eye that is referred to by the colloquial terms "white of the eye".

2nd section: The middle or uveal layers of the eye is densely pigmented, well supplied with blood, and includes three major structures—the iris, the ciliary body, and the choroid:

- **Iris:** it is a circular, adjustable diaphragm with a central hole (the **pupil**), sited in the anterior chamber behind the cornea. It regulates the amount of light entering the eye by adjusting the size of the pupil. It also gives the eye its color, which varies depending on the amount of pigment present.

- **Ciliaray Muscle:** It is a ring-shaped muscle attached to the iris. It is important because contraction and relaxation of the ciliary muscle controls the shape of the lens

- **Choroid:** It is a thin membrane located behind the retina and is connected to the posterior section of the ciliary body. It has a good blood supply and provides oxygen and nutrients to the retina, it also absorbs unused radiation.

The front of the eye houses the anterior cavity which is subdivided by the iris into the anterior and posterior chambers:

- **Anterior chamber:** It is the region of the eye between the cornea and the lens that contains aqueous humor.

- **Aqueous humor:** It is a clear watery fluid located in the anterior chamber of the eye supplying nutrients to the lens and cornea, and disposing of the eye's metabolic waste.

- **Posterior chamber:** It lies behind the iris and in front of the lens. The aqueous humor forms in this chamber and flows forward to the anterior chamber through the pupil.

3rd section: The posterior cavity is lined entirely by the retina, occupies 60% of the human eye, and is filled with a clear gel-like substance called vitreous humor:

- **Vitreous humor:** It is a clear jelly-like substance that fills the eye from the lens on back. It consists of 99% water, contains no cells, and helps to maintain the shape of the eye and support its internal components.

- **Hyaloid:** is a transparent membrane that divides the aqueous humour from the vitreous humour.

- **Lens:** It is a crystal-clear, transparent ellipse-shaped body. The entire surface is smooth and shiny, contains no blood vessels, and is encased in an elastic membrane. It is located in the posterior chamber behind the iris and in front of the vitreous humor. It helps to focus light on the retina.

- **Retina:** The light-sensitive layer of tissue that lines the back of the eye. The retina may be described as the "screen" on which an image is formed by light that has passed into the eye. It contains photosensitive elements (called rods and cones) that convert the light they detect into nerve impulses that are then sent onto the brain along the optic nerve.

- **Optic Nerve:** the bundle of nerve fibers that carry visual messages from the retina to the brain.

Ophthalmic pathologies tend to produce various alterations to the eye's physical properties (i.e. thicknesses of interfaces), you may view **Appendix I** for more details.

1.4 A-Scan Systems

Amplitude-mode (A-mode) ultrasound is used to judge the depth of an organ, or otherwise assess an organ's dimensions. When the ultrasound beam encounters an anatomic boundary, the received sound impulse is processed to appear as a vertical reflection of a point. On the display, it looks like spikes of different heights (hence, the amplitude) [1].

The A-scan provides a one-dimensional image of spikes or deflections of varying amplitude along a baseline, because of the amplification design, differentiation of tissue is possible based on the reflectivity (height of spikes) and structure (distribution of spikes) produced by the cells of various tissue [2].

The height of the vertical reflection corresponds to the intensity of the returning impulse (size of the echo), while the position of the spike along the horizontal axis indicates the time of receiving the echo. The distance between these spikes can be measured accurately by dividing the speed of sound in different tissues by half the sound travel time. Each reflected echo represents a return transit through the material so that the acoustic path length is equal to twice the thickness of the sample.

The sound wave is transmitted and received by the same transducer through a probe that is placed directly on the eye, reflected signals are returned to the probe, mechanical energy is converted to electrical energy, and signals or echoes are recorded on an oscilloscope. The height of the spike indicates the size of the echo.

1.4.1 A-Scan Uses

The most common use of the A-scan is to determine the eye length for calculation of IOL, where data on the length of the eye is a major determinant in common sight disorders. A typical A-scan system measures the following:

- **Biometry Axial Length Measurements**
- **The anterior chamber depth (ACD):** is measured between the anterior corneal surface and the anterior lens surface
- **The lens thickness:** is measured between the anterior lens surface and the posterior lens surface.
- **The Vitreous Cavity's Depth (VCD): i**s measured between the posterior lens surface and the anterior surface of the retina.
- **IOL:** An intraocular lens is an implanted lens in the eye, usually replacing the existing crystalline lens because it has been clouded over by a cataract, or as a form of refractive surgery to change the eye's optical/refractive power-based on the readings of the axial length, the program automatically suggests the choice for intraocular lens strengths. The IOL power is adjusted according to the desired postoperative refraction.

1.4.2 Scanning Techniques

A problem of selecting the appropriate scanning technique involves a trade-off between accuracy of measurements and patient comfort. There are two main techniques:

1.4.2.1 Applanation/Contact Biometry

- **Typical contact biometry**:

Ultrasound probe is placed directly on the cornea, which slightly indents the surface generating the first echospike [3],[4]. The five basic limitations of applanation A-scan biometry are:

  - Variable corneal compression.
  - Broad sound beam without precise localization.
  - Limited resolution.
  - Incorrect assumptions regarding sound velocity.
  - Potential for incorrect measurement distance.

- **Developed Contact biometry:**

Handheld probes cause a wide variance in measurements and impose random errors highly dependent on multiple users of the system and moreover on the grasp and fine control of each individually. Abolishing this inconsistency will be in benefit of precise results. Proposed refinement to the currently known Contact technique rendering it to be more accurate to the eye measurements will act as the ultimate solution for more appreciated system [5].

The patient is examined in the seated position, with the chin correctly positioned on a free-standing chin rest. The ultrasound probe is attached to a zero-weight balance glide to prevent any pressure on the eye during examination, consequently eliminating the corneal compression error. The probe's movements are controlled through a joystick handle as in the figure below. The probe is then brought forward to gently touch the cornea. This technique can only be applied using contact techniques as immersion techniques would contradict with the positioning of the patient.

Figure 1.4-1 A-Scan examination

1.4.2.2 Immersion Technique

The ultrasound probe does not come into direct contact with the cornea, but instead uses a coupling fluid between it and the probe, preventing compression. The patient reclines and a small scleral shell is placed on the anesthetized limbus. Theses shells are placed between the eyelids and centered over the cornea. The probe is either hand-held within the fluid, or locked into position in an infusion shell. The contact lens saline fluid is infused into the shell [3],[4].

When the probe is aligned along the visual axis (center of the macula), five tall spikes are clearly visible in the scan (representing echoes from the cornea, anterior lens, posterior lens, retina and sclera) with the retinal spike rising steeply from baseline. The corneal spike is separate from the probe spike and has a double peak, representing the epithelial and endothelial layers of the cornea.

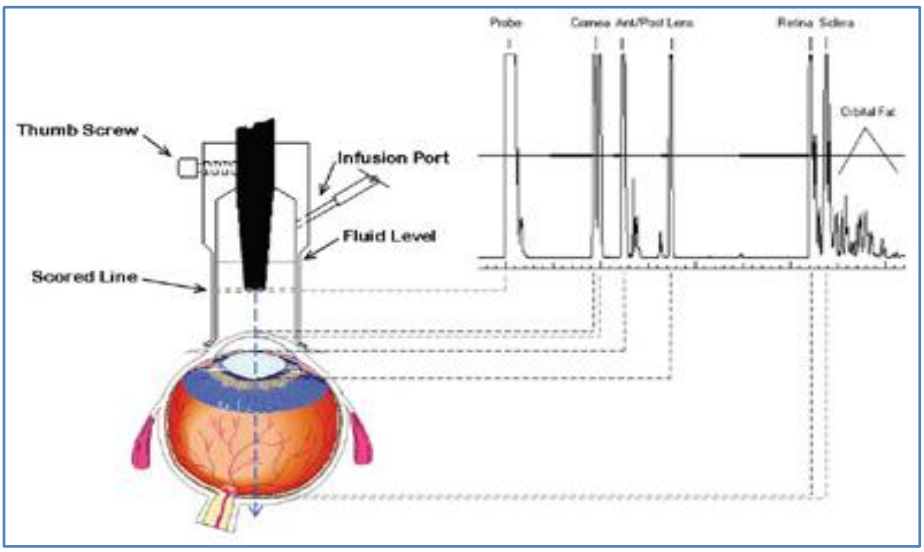


Figure 1.4 - 2 San results

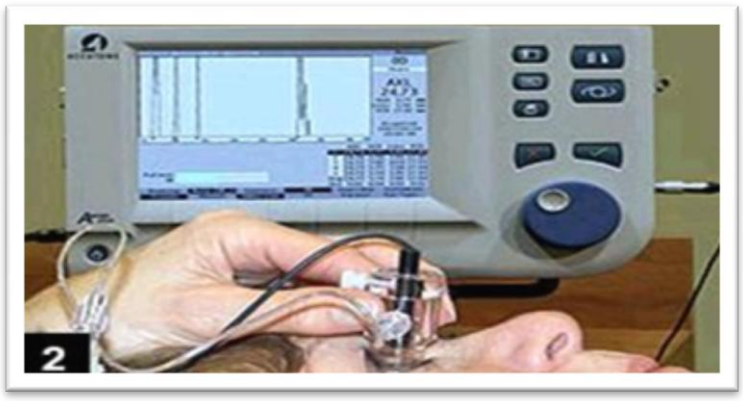


Figure 1.4 - 3 A-scan examination and resulting spikes

Although the immersion technique yields consistent and reproducible measurements, we preferred to apply the contact technique as it much more comfortable for patients.

1.4.3 Eye Modes

There are several patient eye modes each is discussed below**:**

- **Phakic:** is the presence of the natural crystalline lens.

- **Aphakic:**
    - **No lens:** is the absence of the natural crystalline lens, either from natural causes or because it has been removed
    - **Intact:** where only the anterior lens is absent

- **Pseudo-phakic:** When measuring eyes that have previously undergone cataract surgery, where the natural crystalline lens is substituted by a synthetic lens. The tissue velocity must be corrected for the implanted IOL (i.e. for PMMA IOL is 2718 m/s, for Acrylic IOL is 2120 m/s and for Silicon IOL is 1049 m/s). If the user is not sure what kind of IOL was implanted, an observation of the retinal echo distance may help.

- **Dense Cataracts:** In a dense cataract, additional low echoes might appear between the Anterior Lens and the Posterior Lens echoes. Some reverberations of the Anterior and/or posterior Lens echoes might be present in the vitreous when the cataract is very dense especially when higher Gain is needed in order to capture the retinal echo. In a very dense cataract the user may need to increase the gain setting in order to achieve good retinal echo. This is due to the higher absorption of sound in the dense lens.

1.4.4 Output Spikes

When the probe is aligned along the visual axis (center of the macula), six tall spikes are clearly visible in the scan representing echoes from the anterior cornea, posterior cornea, anterior lens, posterior lens, retina and sclera, with the retinal spike rising steeply from baseline. When using the contact mode, the anterior corneal spike bonds with the spike from the probe producing a single spike, representing the epithelial and endothelial layers of the cornea.
Perpendicularity is achieved when all spikes are of high amplitude and the retinal spike is steeply rising from the baseline. The figure below is an Ultrasound display of the different echo-spikes during contact biometry.

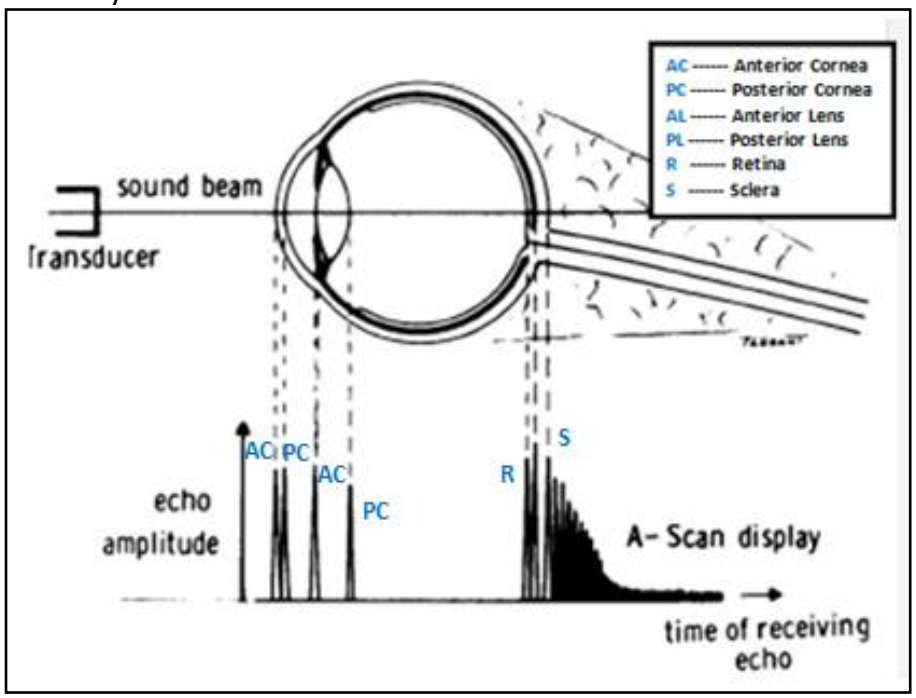


Figure 1.4 - 4 Spikes resulting from interfaces inside the human eye

The Retinal echo should be sharp (90 degree from the baseline) with minimum amplitude height of 70% of the corneal echo. The Anterior Lens and Posterior Lens echoes should also have a height of at least 70% of the corneal echo [6],[7].

- **Normal results:** A normal ultrasound scan would indicate a fully healthy eye. For therapeutic ultrasound, a normal result would be an improvement in the targeted condition, such as shrinking of a tumor or lessening of pressure inside the eye of a glaucoma patient.

- **Abnormal results:** Because diagnostic ultrasound is generally used to investigate symptoms, the results of a scan will often be abnormal and they will detect evidence of an underlying condition.

1.5 Market Survey Statistical Results

1.5.1 A-Scan Systems Market Survey:

We carried out an intensive market survey of standard A-scan Biometry systems from various approved products manufactured by international ultrasound companies, A-scan systems under research, to study the following [8]:

- Typical and advanced features
- Estimate price
- Utilized probe (A and/or B)
- Type of applied scanning technique (Immersion and/or contact)
- Applied frequency range
- Used IOL formulae

Also, we made an A-Scan probe-survey to purchase one for our system as constructing an ultrasound probe would be extremely difficult due to the housing and the assembly. You may view **Appendix II** for more details on the Market Survey. The extracted results are demonstrated in the chart below, showing the weight of each of the Ultrasound Biometry Systems constituent:

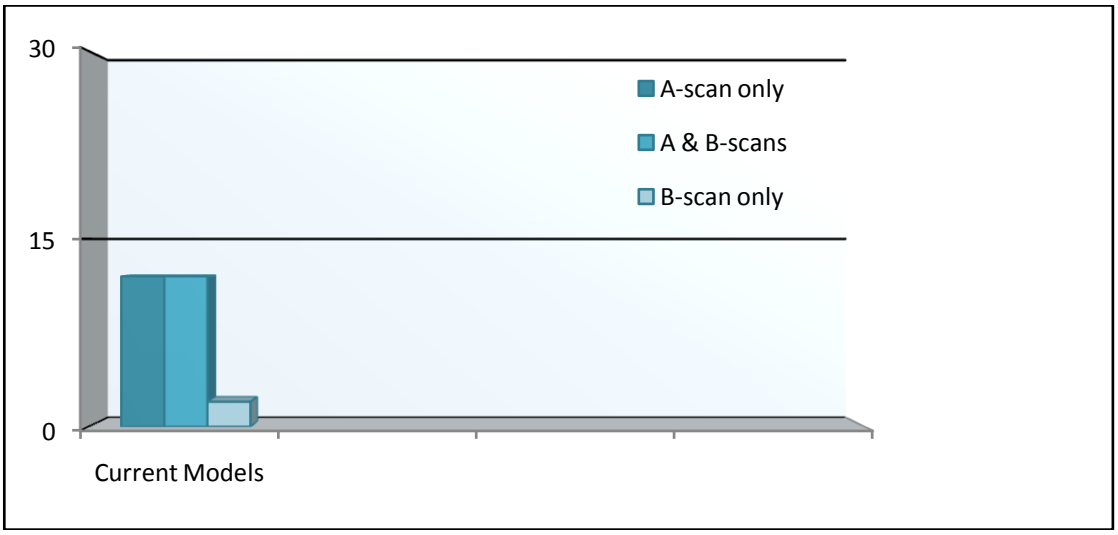


Figure 1.5 - 1 Statistical Results for scanning techniques

1.5.2 Ophthalmologists Survey:

We also managed to survey various ophthalmologists to determine the type of common A-Scan systems used, in addition to inquire any desired features that would be added to the typical A-scan systems.

Our conclusion of the commonly-used A-scan Systems is demonstrated in the graph below, showing that almost all ophthalmologists would prefer to purchase a Stand-Alone A-scan assembly.

1.5.2.1 Commonly-used A-Scan Systems:

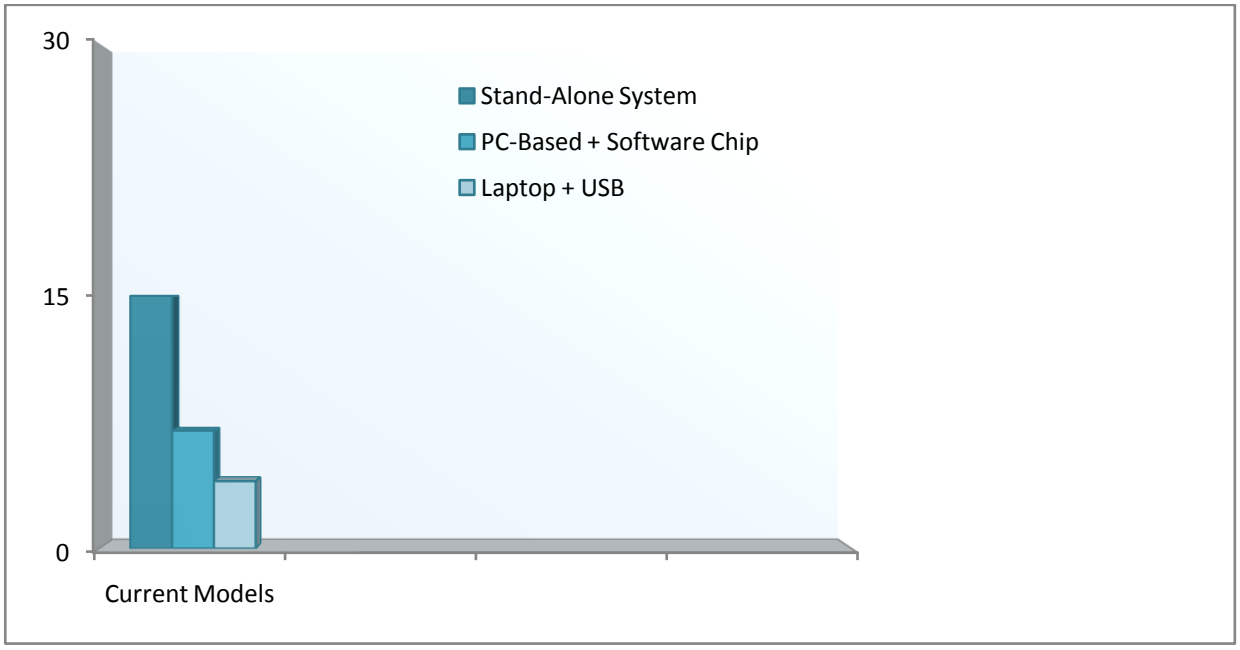


Figure 1.5 - 2 Statistical Results for A-Scan system types

1.5.2.2. Preferred A-Scan Systems Design:

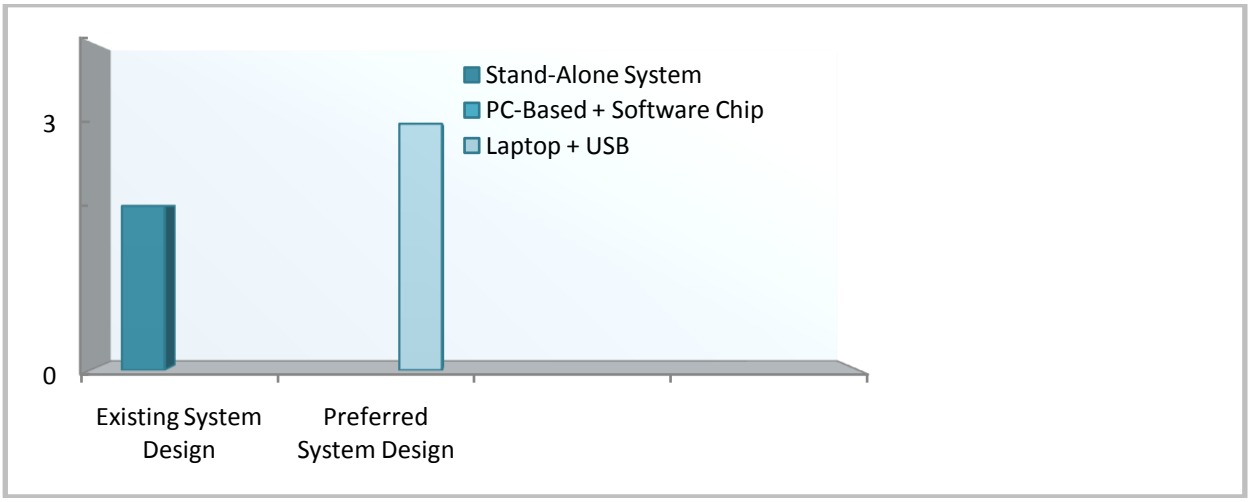


Figure 1.5 - 3 Statistical Results preferred system design

1.6 Work Sequence and Design Flow

The implementation of a complete Enhanced A-scan Biometry system requires a precise, well pre-defined design flow incorporating all necessary stages that guarantee integrity and reliability of the system. Our work progression was pipelined by the block-diagram shown in Figure 1.6 - 1. [9]

Having a clear vision of the added value and objective of our system we started to list all its requirements and specifications. This is shown in **Chapter 2**. The Development of algorithm as well as its simulation with a high level DSP tool is performed in **Chapter 3**. Moving from the Simulation level to the implementation level is demonstrated in **Chapter 4** by first identifying our system's components and wisely selecting the devices that are most appropriate to our application. The Hybrid Model of the enhanced A-Scan system is presented here and thus **Part I** terminates with the grounding for the implementation. Execution of the design is illustrated in **Part II**, where branching into the Software and Hardware units is revealed. Detailed rationalization is provided in **Chapters 5** through **8**. Integration of the components and validation of their functionality is presented in **Part III**. In **Chapters 11** to **12** of **Part IV** we discuss the results and limitations of our design. Also the foreseen future work is presented in this part.

Many Appendices wrap up the design and provide all necessary basic knowledge that support comprehension of the underlying enhanced A-Scan biometry system.

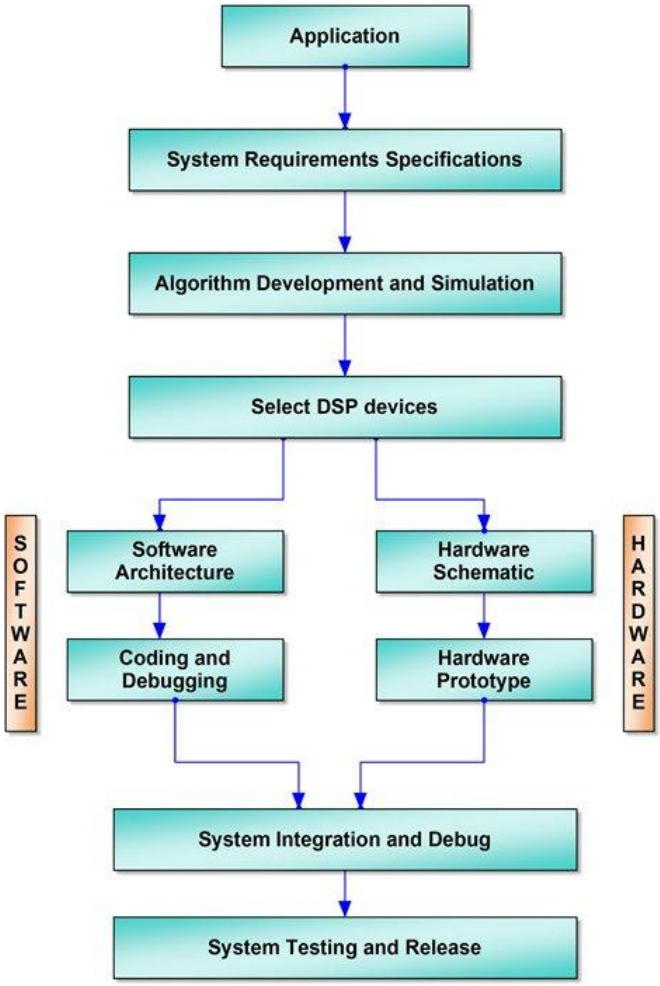


Figure 1.6 - 1 Design Flow

1.7 References

[1] William R. Hendee & E. Russell Ritenour, Medical Imaging Physics, Wiley-Liss Inc., 2002
[2] http://www.eyetec.net/ce/M1S3.htm
[3] http://www.emedicine.com/proc/TOPIC486.HTM
[4] http://www.freshgasflow.com/physics/us_sound/ultrasound.html#introscan_a_scan
[5] http://folk.ntnu.no/stoylen/strainrate/Ultrasound/
[6] S. W.LEGE & B. MILLER & SCHNEIDER, A-Scan Systems, 2000
[7] Jacques Poujolg, Echography in Ophthalmologyn, Masson Publishing, 1985
[8] http://www.jultrasoundmed.org/cgi/reprint/26/2/149
[9] Richard G. Lyons , Understanding Digital Signal Processing, Prentice Hall, 2001

Chapter

2

System Requirments Specifications

2.1 Introduction

After formulating the problem with as much specificity as possible and determining required output it is now time to put hands on the specifications and requirements needed to achieve the desired output (Figure 1.6-1). We then creatively synthesize the requirements into design solutions believed to satisfy those requirements. The set of decision-making processes and functional blocks that make up the system are exhaustively elucidated by detailed block diagrams and accompanying explanation.

2.2 System Features & Specifications:

2.2.1 Features

We propose an enhanced relatively-low-cost A-scan Biometry system which connects via USB cable to any Windows XP laptop or desktop computer. All features are thread below in details:

Standard Features:

- Automatic and Manual modes
- Contact settings
- Axial length, lens thickness, vitreous depth, and anterior chamber depth of the eye are to be measured accurately abiding to the standards of A-scan Biometry systems worldwide standards.
- Accurate eye dimension measurements and IOL power calculation.
- Software includes all IOL formulae (SKR, SKR II, SKR-T, BINKHORST, HOLLADAY, HOFFER-Q, HAIGIS).
- Lens correction factor
- Continuous plotting of acquired signal
- Freezing accepted signal and continuing scans
- Saving patient information and images
- Facility to print a full report
- Amplification (Fixed and TGC)
- Thresholding to eliminate noise

Added Features:

- Aiding in Automatic Diagnosis of any abnormalities.
- Enhanced Contact Technique
- Automatic Detection Software selects the best scans (by calculating average)
- Post-processing enables optimization of A-Scan Gain and TGC after the patient examination (adjustable TGC)
- Adjustable velocity according to lens type
- Probe Misalignment check
- Guiding error messages for misalignment and corneal compression
- Labeling peaks using gates
- Calculating standard deviation and average of gated signals
- Manual and automatic selection of IOL formula
- Comparison between 2 or more formulae for IOL calculation

- Manual labeling of unrecognized signal (semi automatic)
- User input of observed case
- TGC adjustment in manual mode
- Dense cataract modification of gain
- Adjustment of reading for Traumatized eye
- Ribbon display of all valid scans in automatic mode and their average.
- Processing based on 3 different modes, fully automated, manual and semi manual (manual labeling)
- Peak detection implementing 2 techniques, peak position at maximum and calc area under envelope for peak amplitude
- System controlled by various user controls sent via UART
- Calculation of avg and standard diviation of all peaks
- Compact

2.2.2 Specifications

2.2.2.1Physical description

a. Dimensions
   i. 165 x 93 x 188 mm
   ii. weight: 0.8kg

b. Materials
   i. Biometry Probe – body: GE Noryl EN265
   ii. Housing – ABS, UL 94 V0

c. Power Supply
   i. Input Voltage range:
   ii. Output Voltage: 25,-15,15,5,3.2,
   iii. Frequency: 50/60Hz
   iv. AC power consumption: 10VA @ 120V
   v. DC power consumption: 4.2W

d. Operating conditions
   i. Temperature:
   ii. Operating = 19 degrees to 35 degrees C (32 to 95 F);
   iii. Storage= -40 to 65 C (-40 to 149 F)
   iv. Relative humidity:
   v. Operating = 10% to 90% (non-condensing);
   vi. Storage: 5% to 95% (non-condensing)

e. 3D reconstruction
   i. Scan time: 26.70 ms
   ii. Angular resolution: < 2
   iii. Capture depth: 44mm
   iv. Display resolution: 0.1mm on each axis

2.2.2.2 Measurement Accuracy

a. Measurement accuracy is limited by the technique employed.
b. Maximum 0.12mm, assuming no errors in technique

2.2.2.3 A- Scan

a. Reference: US-PRO-B
b. Frequency: 9MHz
c. Operating mode: Pulsed
d. Active diameter: 7mm
e. Active surface: 154mm2
f. Acquisition
   ii. Horizontal resolution: 1024 points max
   iii. Vertical resolution: 256 lines
   iv. Linear resolution: 0.052mm

2.3 System Block Diagram

2.3.1 Higher-Level Block Diagram

Our A-Scan ultrasound system is an ultrasonic diagnostic system intended to be used for ophthalmic applications. Scans can be made by placing the probe directly on the eye (hence, contact technique).

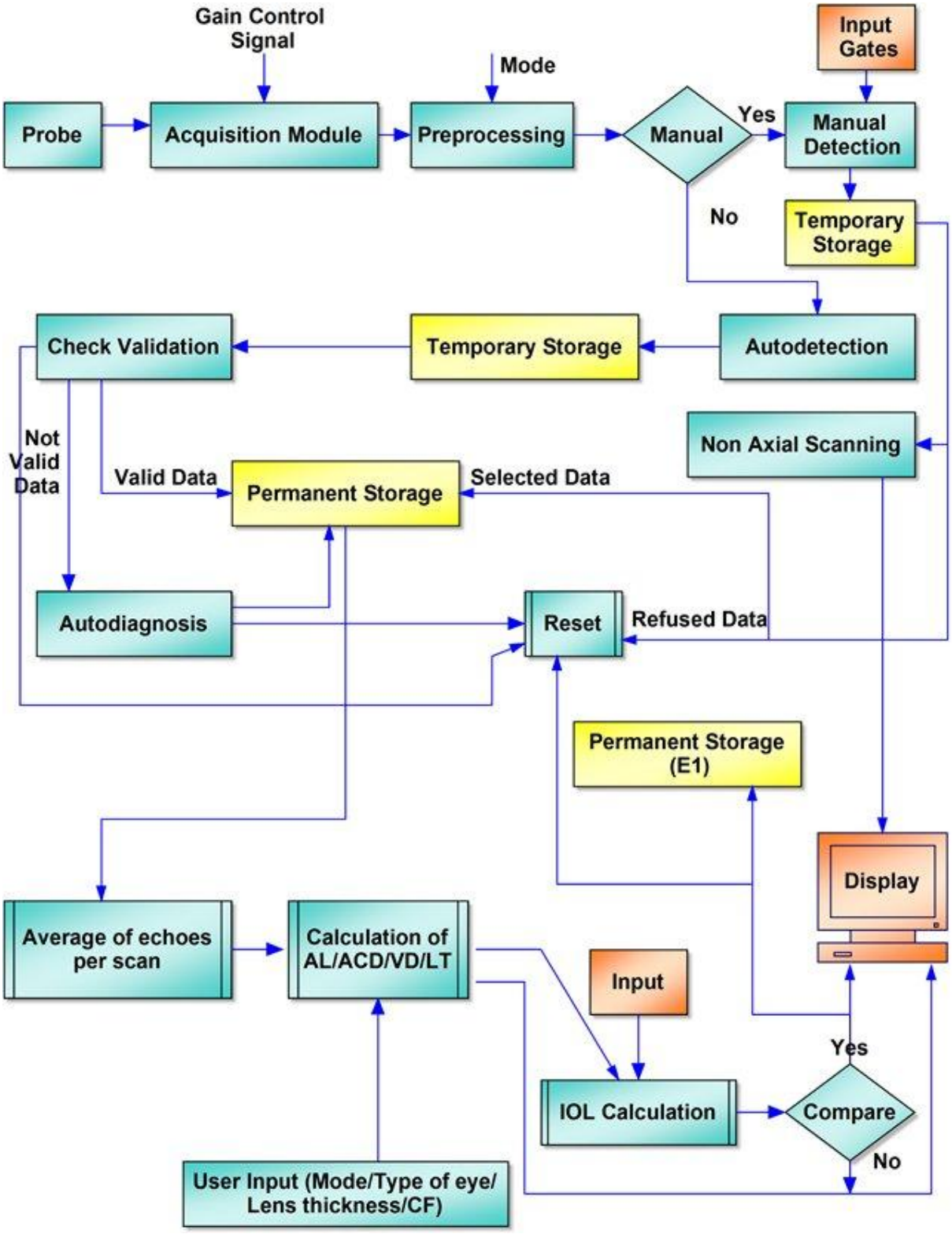


Figure 2.3-1 High-Level Block Diagram of e-DIOLA scan

2.3.2 Lower-Level Block Diagram

The e-DIOLA scan system consists of an ultrasound unit, which contains the ultrasonic pulsing and receiving circuitry and scan converter, and an attached computer for control, patient data storage and calculations.

Main Blocks

1. Signal Acquisition
2. Preprocessing — Amplification Phase; Thresholding
3. Mode Selection — Manual detection & Input Time Gates; Autodetection & Check Signal Validation
4. Eye Calculation
5. IOL measurements
6. Display
7. Auto-diagnosis
8. Angular probe Algorithm

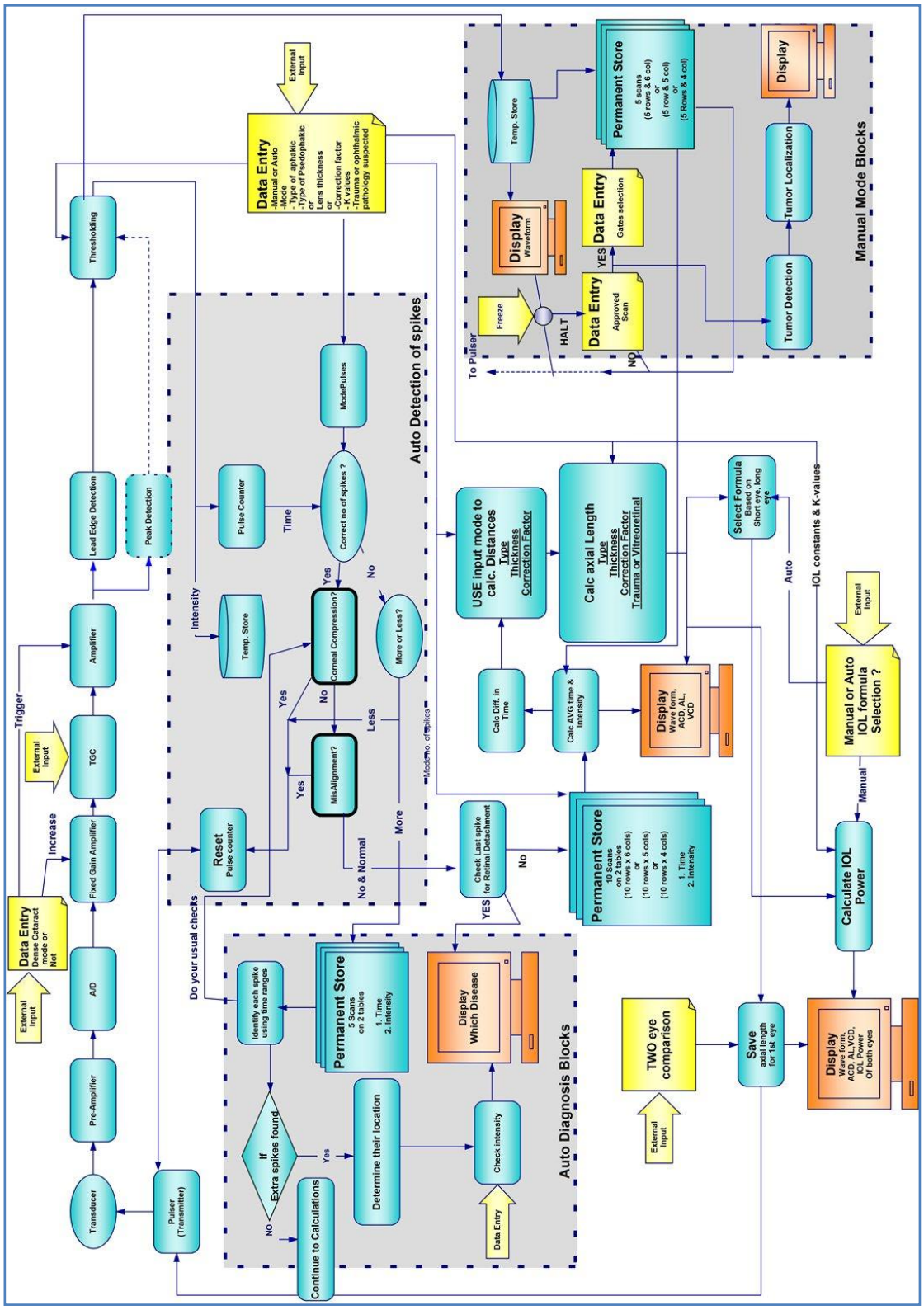


Figure 2.3-2 Low-Level Block Diagram of e-DIOLA scan

2.3.2.1Signal Acquisition Module

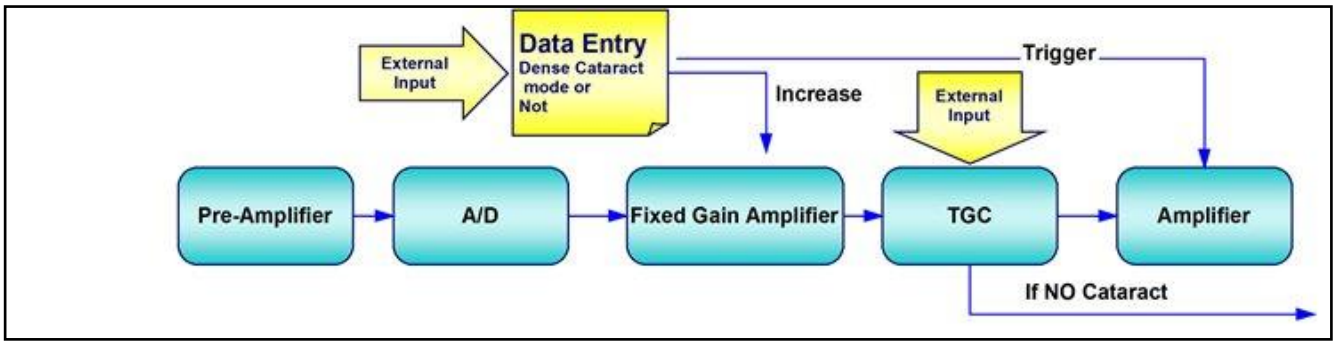


Figure 2.3-3 Signal Acquisition Module

- **Input:** Signal from probe
- **Operation:**

The ophthalmologist is only required to adjust the device to the patient's eye and initiate the measurement. The system will automatically decide when we have a good scan unlike the manual mode where the operator is allowed to decide.
A typical older machine has an automatic mode that simply grabs any spike that ventures over a threshold value in a prescribed area along the baseline. Whereas our automatic mode will be built on a well-based analysis in identifying each spike using more sophisticated pattern recognition algorithms.
For greater accuracy, measurements are taken from the position of intersection of the leading edge of the pulse with the time base-line and not that of the spike.

The Subsystems of this module are:

1. **Preamplifier:** for impedance matching the signal and prevent ion of signal loss

2. **ADC :** digitizes the analog received signal

3. **Fixed gain Amplifier:** amplifies the signal by a fixed amount of gain.

4. **Time Gain Compensation (TGC):**

The attenuation of the ultrasound signal by the different interfaces of the eye is usually considered an undesirable factor because it produces falloff of signal intensity with depth without yielding any useful information [2]. This attenuation can be compensated for by use of Time Gain Compensation (TGC) circuit. The key to compensating for this falloff is the relationship between the depth and time of arrival of the echo at the transducer. Echoes returning at longer times receive greater amplification to compensate for increased attenuation by different interfaces of the eye.

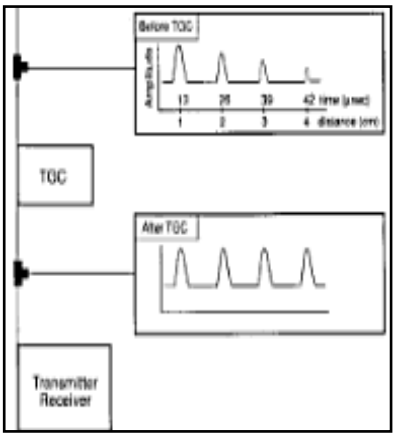


Figure 2.3-4 TGC effect illustration

There are two common types of TGC controls:

1. A series of linear potentiometers for discrete setting of time delays at various depths.
2. A three knob control that allows adjustement of initial gain, slope and far gain.

When using higher frequency, the ultrasound beam attenuates more due to the dispersive absorption of the tissue. Thus, the slope at higher frequencies is larger than that with lower ones.

Typical attenuation for soft tissue → 1dB/cm MHz; 1dB/MHz for interval of time corresponding to 1cm of pulse travel.

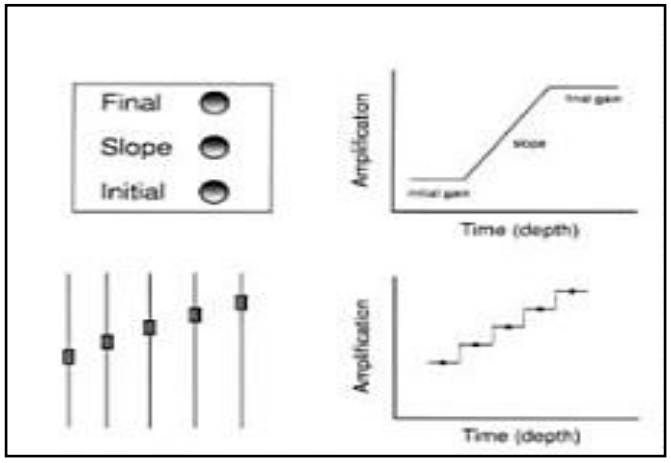


Figure 2.3-5 Types o f TGC controls; Top: Type 1; Bottom: Type2

Corneal spike will have two spikes corresponding to the epithelium and endothelium. Care should be taken to keep the gain low enough to appreciate and resolve these two spikes. If the gain is set too high, poor resolution of these two interfaces will occur and the corneal spike will appear wide and flattened.

- **Output:** Amplified signal

2.3.2.2 Pre-processing Module

- **Input:** Amplified Signal
- **Operation:**

In this stage all spikes having amplitudes below the valid threshold value are eliminated. This is called thresholding. In order to select the threshold value, we must determine highest spike of all the eye interfaces. Basically, the two corneal spikes tend to constantly have the highest amplitude.

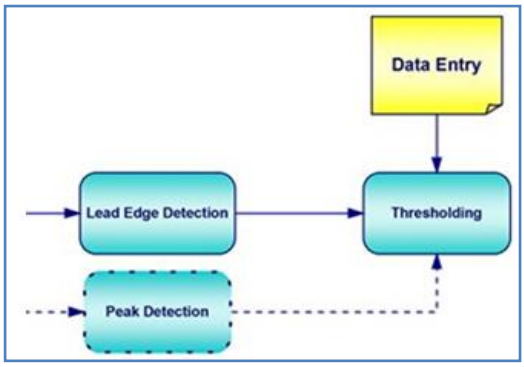


Figure 2.3-6 Thresholding Module

Measuring lights are attached to spikes A, C, and D because they are the first three spikes (beyond the probe/cornea spike) to venture above the threshold line (T). Spike B is ignored because it is below the threshold line.

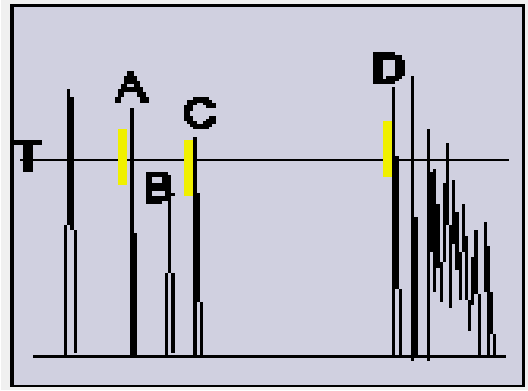


Figure 2.3-7 Thresholding illustration

- **Output:** Thresholded signal

2.3.2.3Mode Selection Module

A. Autodetection & Signal Validation:

- **Input:** Spikes from eye Interfaces

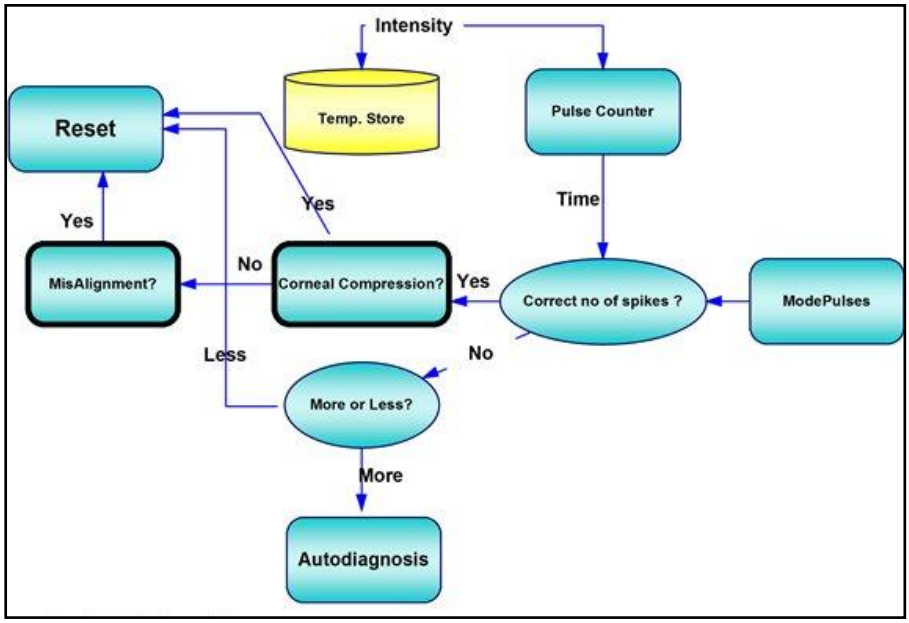


Figure 2.3-8 Autodetection Module

- **Operation:**

Auto-detection: Selecting the Best Scans.Rreviewing each scan; and verifying the position of the 6 markers. Storing multiple measurements and calculating their average. Calculating other statistical data such as the standard deviation
Corneal compression recognition: monitoring anterior chamber depth by automatically rejecting measurements less than the appropriate axial length in automatic mode.

Check Validation: Probe Misalignment (PM) and Corneal Compression (CC) are two common errors due to the manual handling of the probe during scans. The DIOLA software system tends to alert the ophthalmologist when these errors occur to begin a new scan.

Misalignment: If either the AL or Retinal echo is too low, it is an indication that the probe is off the visual axis. The axial length may be either too short or too long.

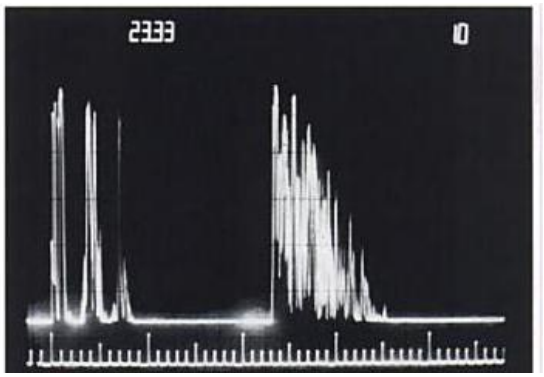


Figure 2.3-9 A-scan display of the same eye with an on-axis measurement of 23.33 mm.

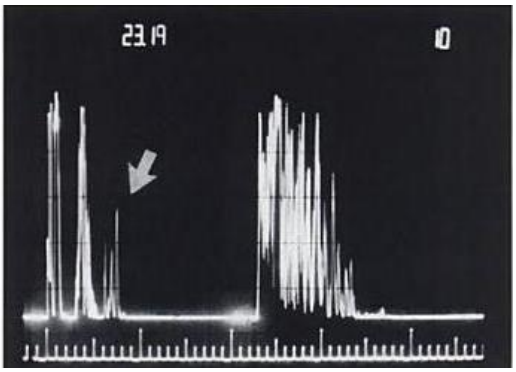


Figure 2.3-10 Slight Off-axis measurement of 23.19 mm. Note the presence of the small posterior lens surface echo-spike (arrow).

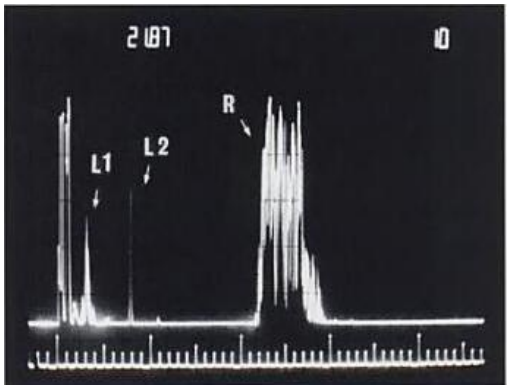


Figure 2.3-11 Large Off-axis measurement of 21.87 mm. Note the presence of the small lens spikes (L1 & L2) and the jagged retinal spike (R).

Corneal compression recognition: recognizing when the probe is indenting the cornea by monitoring anterior chamber depth by automatically rejecting measurements less than the appropriate axial length in automatic mode in the automatic mode.

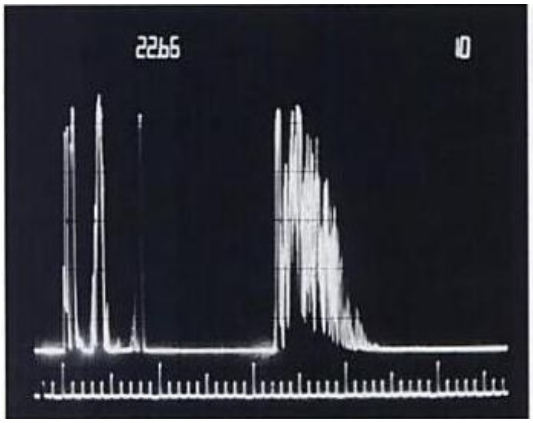


Figure 2.3-12 Phakic eye with Corneal Compression showing an axial length reading of 22.65 mm.

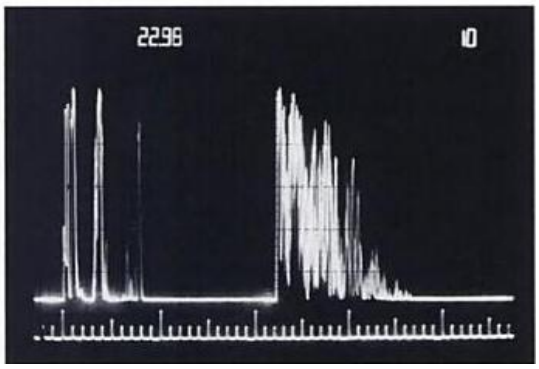


Figure 2.3-13 Phakic eye without Corneal Compression showing an axial length reading of 22.96 mm.

- **Output:** identified peaks, while:
  Valid Spikes are sent to Eye Measurement Module
  Abnormal Spikes are sent to Auto-diagnosis
  Invalid Spikes are sent to the Reset to rescan.

B. Manual Mode & Input Time Gates:

- **Manual Peak Detection:**
  - **Input: Spikes from eye Interfaces**
  - **Operation:**

The ophthalmologist selects the echoes by simply placing a marker at the top of the peaks representing the 6 main interfaces of the eye.

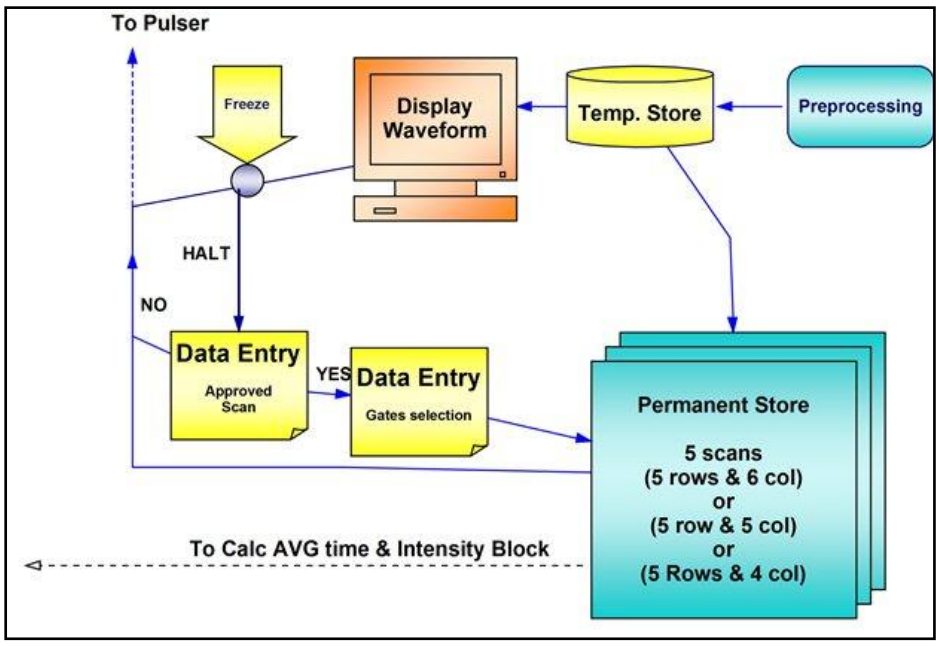


Figure 2.3-12.3-1 Manual Peak Detection Module

- **Input:** Time Gate Protocol
    - Employing a five-gate measuring system.
    - Biometery does individual velocity measurements for the aqueous, the lens, and the vitreous.
    - first gate measures position of cornea,
    - second measures position of anterior lens spike,
    - third measures position of posterior lens spike,
    - Fourth measures the position of retina spike.
    - Units have visible and moveable gates.
- **Output:** identified peaks

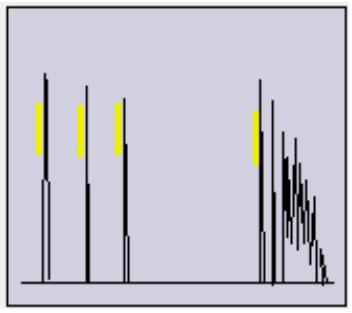


Figure 2.3-15 Yellow markings on displayed signal are placed by the ophthalmologists guided by Input Time Gates to identify the main eye spikes.

2.3.2.4 Eye Calculations Module

- **Input: Time & Amplitude Arrays from Manual or Automatic modes**
- **Operation:**

Many A-scan instruments do not measure each individual segment in the eye. Instead they use an average velocity of sound. Many use 1550 for a phakic eye (natural lens) and 1532 for an aphakic eye.

- **Scan Capture:**
    - Capturing at least 6-10 measurements (scans)
    - Selecting at least 3–4 good scans with an Axial Length standard deviation (SDAL) of 0.1 or lower should be saved and used for IOL calculations. (The variation of the axial length, from the highest to the lowest, of the scans should be not more than 0.1-0.2mm.)

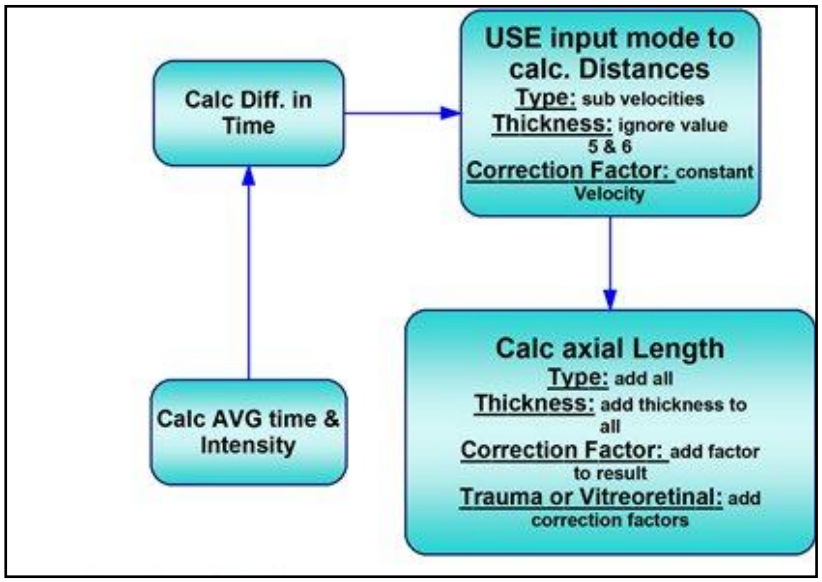


Figure 2.3-16 Eye Calculations Module

- **Axial length (AL) :** The variation of the axial length, from the highest to the lowest, of the scans should be not more than (0.1-0.2) mm

- **Lens thickness (LT):** when measured, use a velocity of 1641 m/s. The sound velocity varies in cataractous eyes with a slower velocity (average 1590 m/s).
- **Anterior Chamber Depth (ACD):** when measured, use a velocity of 1532 m/s. The operator should save the measurements with longest ACD. These are usually the measurements where there was minimum corneal compression. Large variations in AC measurements indicate compression of the cornea and/or off-axis positioning of the probe.
- **Vitreous Cavity's Depth (VCD):** when measured, use a velocity of 1532 m/s.
- **Special Conditions:** approximation of an axial length: the eye to be operated on is difficult to measure, whether it's a recently traumatized eye *or* an eye with vitreo-retinal pathology where the retina cannot be identified. The AL can be approximated by comparing both eyes (hence, opposite eye is measured).

In case the eye under test is traumatized, then the ophthalmologist scans the other healthy one, and the DIOLA software performs the necessary steps to compensate the differences in the two eye measurements despite not scanning the traumatized eye.

- **Output:** Eye Measurements (AL, VCD, LT, VCD)

2.3.2.5IOL Measurement Module

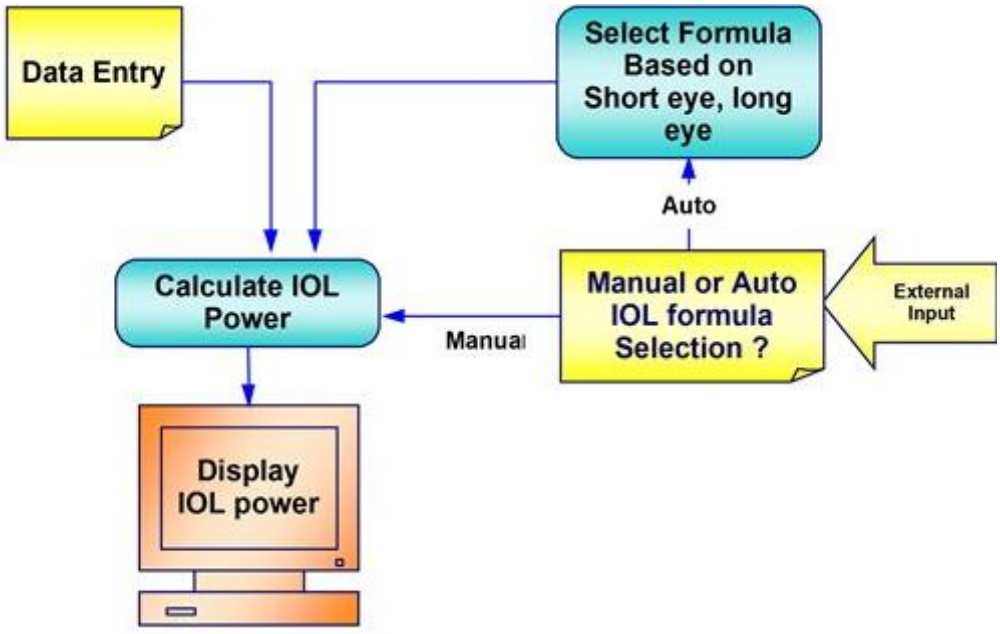


Figure 2.3-17 IOL Measurement Module

- **Input: IOL Formula from Manual or Automatic modes**
- **Operation**:

The new generation of A-scan instrumentation is making IOL measurement easier, faster, and more accurate. They are also building in features that reduce errors and improve accuracy.
Having various built-in IOL formulae, the ophthalmologist selects the most appropriate formula according to the axial length. The most common formula is the "Holladay", you may view **Appendix III for** more details on IOL Formulae.

- **Output:** Calculated IOL Power

2.3.2.6 Display Module

The image below depicts a waveform display. The following are displayed on the system monitor:

- Patient Name
- Date of Test
- Scanning Technique (immersion/Contact)
- Eye Mode(phakic, Aphakic, pseudophakic)
- Axial Length
- ACD
- VCD
- Identified Interface-Peaks

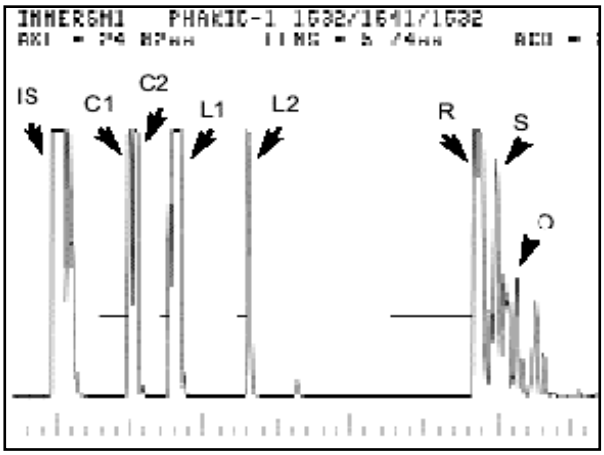


Figure 2.3-18 Model of an A-scan Display Monitor stating the identified peaks. (IS) represents the spike due to the probe interfacing with the eye. (C1) represents Anterior Cornea spike, (C2) represents Posterior Cornea spike, (L1) represents Anterior Lens spike, (L2) represents Posterior Lens spike, (R) represents Retinal spike and (S) represents Scleral spike.

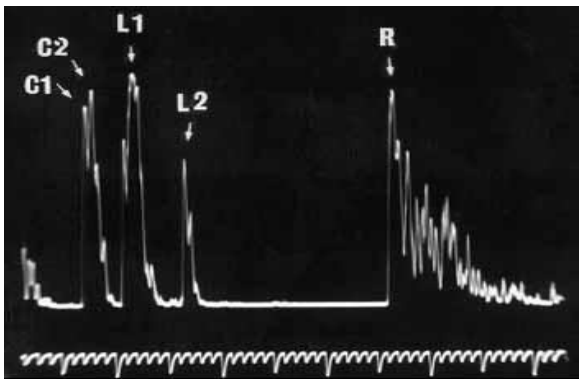


Figure 2.3-19 Another Model of an A-scan Display Monitor stating the identified peaks. (C1) represents Anterior Cornea spike, (C2) represents Posterior Cornea spike, (L1) represents Anterior Lens spike, (L2) represents Posterior Lens spike and (R) represents Retinal spike.

2.3.2.7 Auto-diagnosis Module

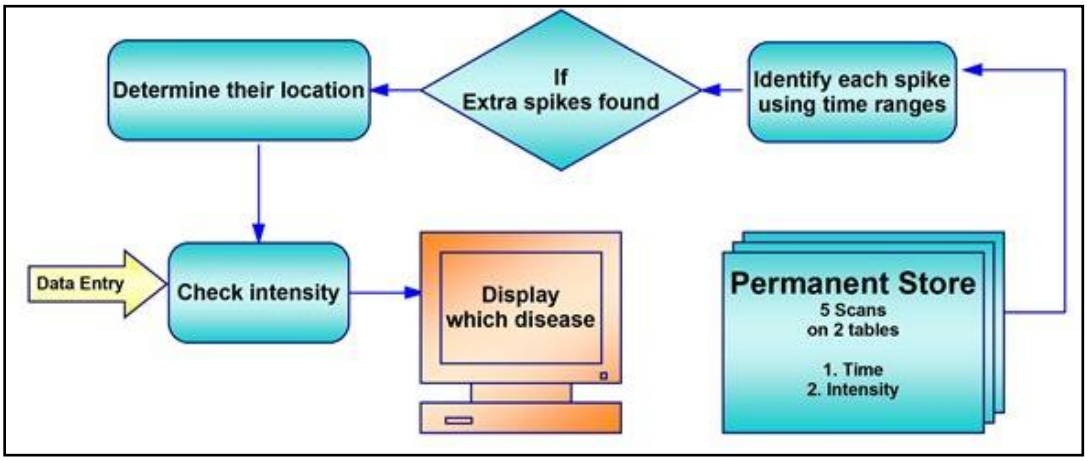


Figure 2.3-20 Eye Auto-Diagnosis Module

- **Input:** Abnormal Spikes from Automatic Mode
- **Operation:**

The auto-diagnostic module is a sub-set of our biometry system. Diagnostic ultrasound is now used routinely in the investigation of patients with opacification of the ocular media or with orbital disorders. Through an intense market survey (Appendix I), only one model (DGH-5000e, DGH Technology INC., Georgia, United States) is provided with feature of auto-detection, discarding faulty readings.
Pathologies that acted as random errors to the outputs of the systems causing an obvious misjudgment of eye dimensions, is to be auto-identified and corrected for more certain results.

As we mentioned before, ophthalmic pathologies tend to produce various alterations to the eye's physical properties (i.e. dimensions of interfaces), you may view **Appendix I** for more details. The nature of abnormalities giving rise to echoes may be suggested by the size, extent and movement of the echoes. Another problem is effect of certain pathological cases (Tumors, and Retinal Detachment) on measurements which have to be detected, interpreted by the ophthalmologist.

Detected echoes may be either point-like or membrane-like, for each of these two types a number of criteria may be used in order to determine the abnormality giving rise to the echoes. Point-like echoes may sometimes be differentiated on echo amplitude and attenuating properties. The acoustic properties of foreign bodies, for example, differ greatly from biological materials; foreign bodies may thus give rise to very high amplitude echoes and may also strongly attenuate the sound beam. In addition, artifacts (e.g. multiple reflections of the sound within the foreign body) may also be produced. Vitreous opacities (e.g. hemorrhage, inflammatory cells or pigment cells) may give rise to echoes with a wide range of amplitudes, and therefore cannot generally be differentiated ultrasonically.
Our auto-diagnosis A-Scan system, we will only be concerned with the alterations made on interface reflectivity (hence, amplitude of generated peak). From the table given below, it is declared that ocular melanoma is the only type of eye tumors that can be detected using A-scan systems only excluding the role of B-scan photography, as it strongly attenuates the US signal due to its low reflectivity [1].

Eye Tumor Types	Acoustic Consistency
Melanoma	Low Reflectivity (Homogeneous)
Secondary	Low Reflectivity (Homogeneous)
Haemangioma	Low Reflectivity (Homogeneous)
Retinoblastoma	Low Reflectivity (Homogeneous)

We managed to collect a lot of abnormal A-scan photographs in order to set the precise criteria of how to discriminate the various ophthalmic pathologies from each other. In addition to any other information related to physical alterations made to the eye due to these pathologies, which the ophthalmologist can load into aid the **DIOLA software** produce accurate diagnosis [1]. The extracted conditions set to identify a particular disorder are illustrated below:

1. **Asteroid Hyalosis:**
   - **Regional Location:** Vitreous space (after the posterior lens spike & before the retinal spike)
   - **Description:**
     - **Mild Case:**
       - Multiple spikes of medium-to-low amplitude
     - **Severe Case:**
       - Multiple spikes of medium-to-high amplitude
       - Physical signs of yellow spots on cornea
   - **A-scan Photograph:**

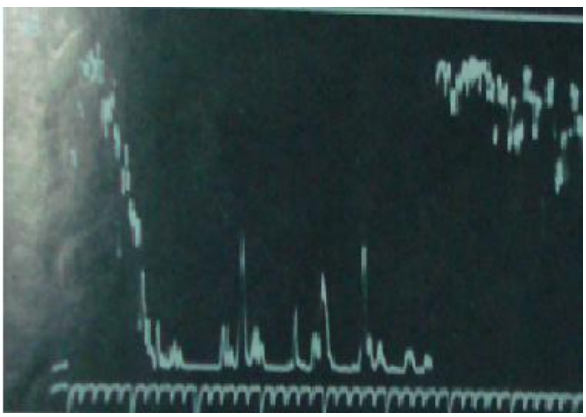


Figure 2.3-21 A few spikes having low amplitude produced by the dispersed asteroid bodies signifying Mild Asteroid Hyalosis.

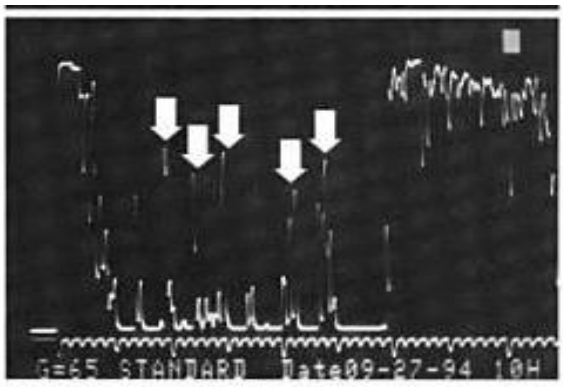


Figure 2.3-22 Another A-scan of Mild Asteroid Hyalosis showing multiple moderately reflective spikes produced by the dispersed asteroid bodies.

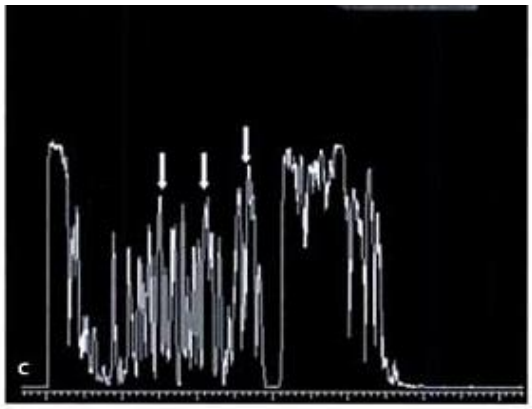


Figure 2.3-23 Severe Asteroid Hyalosis demonstrating multiple highly reflective spikes produced by the dispersed Asteroid bodies

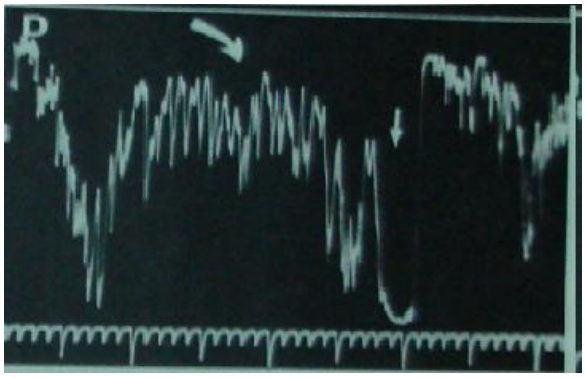


Figure 2.3-24 Multiple spikes of medium-to-high amplitude produced by the dispersed Asteroid bodies (Severe case).

2. **Endophthalmitis (Uveitis):**
   - **Regional Location:** Vitreous space (after the posterior lens spike & before the retinal spike)
   - **Description:**
     - Patient must have undergone surgery
     - Chain of low amplitude spikes
   - **A-scan Photograph:**

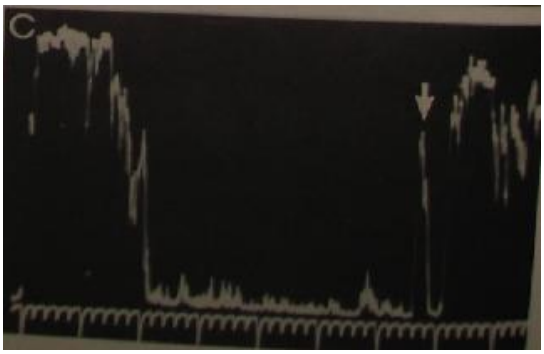


Figure 2.3-25 Extremely low-amplitude spikes produced by Endophthalmitis infection followed by inflammation. The disease is termed Uveitis.

3. **Vitreous Opacities:**
   - **Regional Location:** Vitreous space (after the posterior lens spike & before the retinal spike)
   - **Description:**
     - Few peaks of low amplitude due to the presence of small granular opacities
   - **A-scan Photograph:**

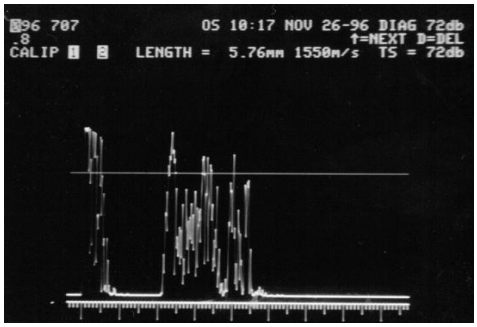


Figure 2.3-26 Normal Vitreous. Notice that the vitreous space doesn't produce any spikes as the other eye interfaces.

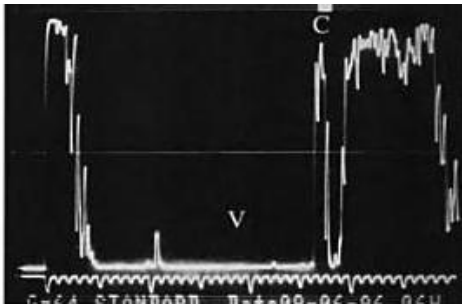


Figure 2.3-27 Very low reflective spikes from the vitreous opacities (V) and a double peaked highly reflective from the shallow detachment(C).

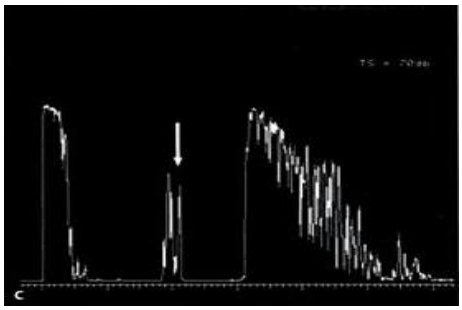


Figure 2.3-28 Multiple moderately reflective spikes produced by a single dense Vitreous Opacity.

4. **Coat's disease:**
   - **Regional Location:** Subretinal space (after the retinal spike & before the sclera spike)
   - **Description:**
     - Multiple spikes of similar amplitude to the main retinal spike
   - **A-scan Photograph:**

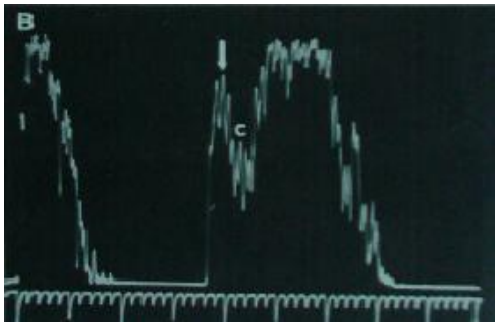


Figure 2.3-29 Coat's Disease, the (arrow) represents a few spikes of similar amplitude to the retinal spike, while (c) are cholesterol bodies.

5. **Ocular Melanoma:**
   - **Regional Location:** Subretinal space (after the retinal spike & before the sclera spike)
   - **Description:**
     - Amplitude of Retinal peak noticeably increases (almost same as corneal spikes), corresponding to the base of the melanoma tumour.
     - While a few spikes of extremely low amplitude corresponding to the core of the melanoma tumour, are observed after.
   - **A-scan Photograph:**

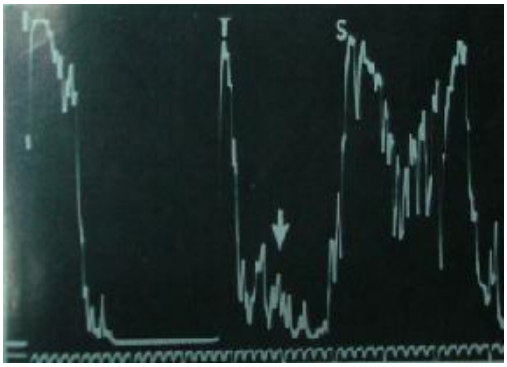


Figure 2.3-30 Ocular Melanoma, where (T) corresponds to the base of the tumour and the (arrow) corresponds to the tumour's core.

6. **Tear Retinal Detachment:**
   - **Regional Location:** Subretinal space (after the retinal spike & before the sclera spike)
   - **Description:**
     - The retinal spike is observed in its expected range but it's slightly indented.
     - The amplitude increases to (100%)

7. **Expulsive Hemorrhage:**
   - **Regional Location:** Scleral space (after the retinal spike & before the sclera spike)
   - **Description:**
     - A chain of multiple decaying spikes right after the retinal spike and before the sclera peak
   - **A-scan Photograph:**

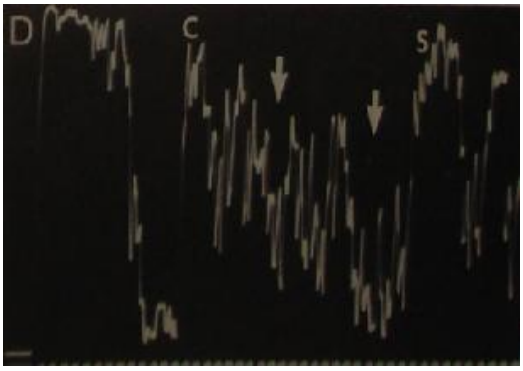


Figure 2.3-31 Expulsive Haemorrhage in the subscleral region after the retinal peak. (C) represents the corneal spike, while the first (arrow) represents a chain of decaying spikes. (S) represents the Scleral Spike.

All the above condition were well studied and the following flowchart was developed for Autodiagnosis.

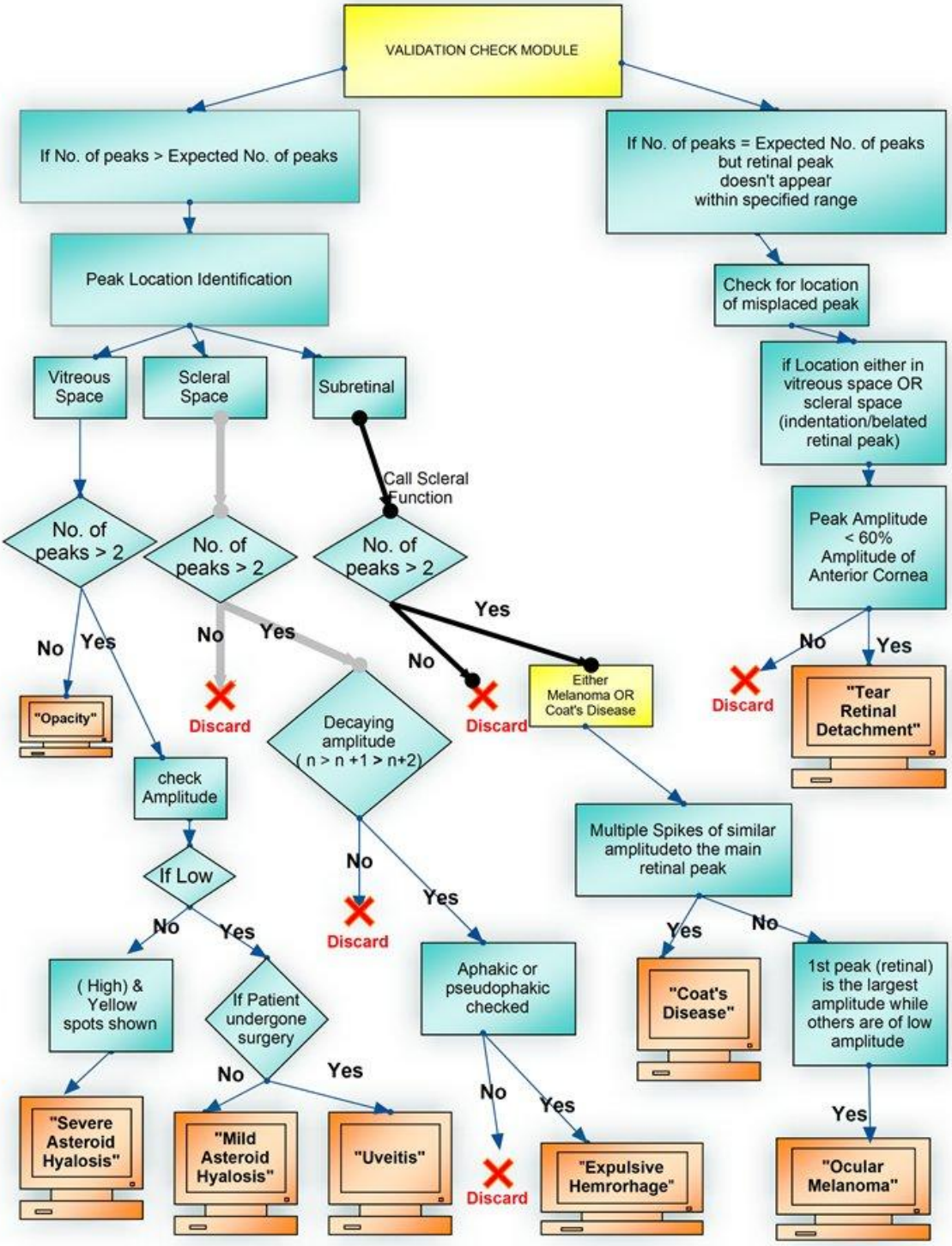


Figure 2.3-32 Eye Auto-Diagnosis Flowchart

- **Output: Pathology Name viewed on display monitor accompanied to the abnormal A-scan**

2.3.2.8 Non-Axial Scanning

- **Input: Spikes from Manual Mode**
- **Operation:**

Over the past few decades, the use of ultrasound in ophthalmology has become an important and often necessary tool to aid in the diagnosis of intraocular and orbital diseases. One of the added values to the typical A-scan systems is detecting irregularities (i.e. tumours) in the orbital cavity (hence, the eye's non-axial plane). In case there is a suspicion of a tumour existing in the orbital cavity, the ophthalmologist may choose to use the introduced feature manually to scan through the entire eye. Thus, the detection of the tumour's position with respect to the axial plane is identified.

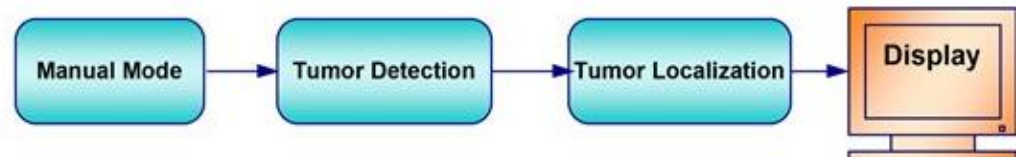


Figure 2.3-33 Non-Axial Scanning Module

- **Algorithm Flowchart:**

The ophthalmologist determines the angle at which the tumour is located by performing two successive scans. The first scan is for calculating the axial length, while the second is used to calculate the ultrasound path length provided that the misalignment check is disabled. We can get the probe's angle-of-inclination by the following equation:

$$\theta = \cos^{-1}\frac{L}{AL}$$

And the radius at which the tumour is located by:

$$R = L \sin(\theta)$$

- **Output:** Tumor Location

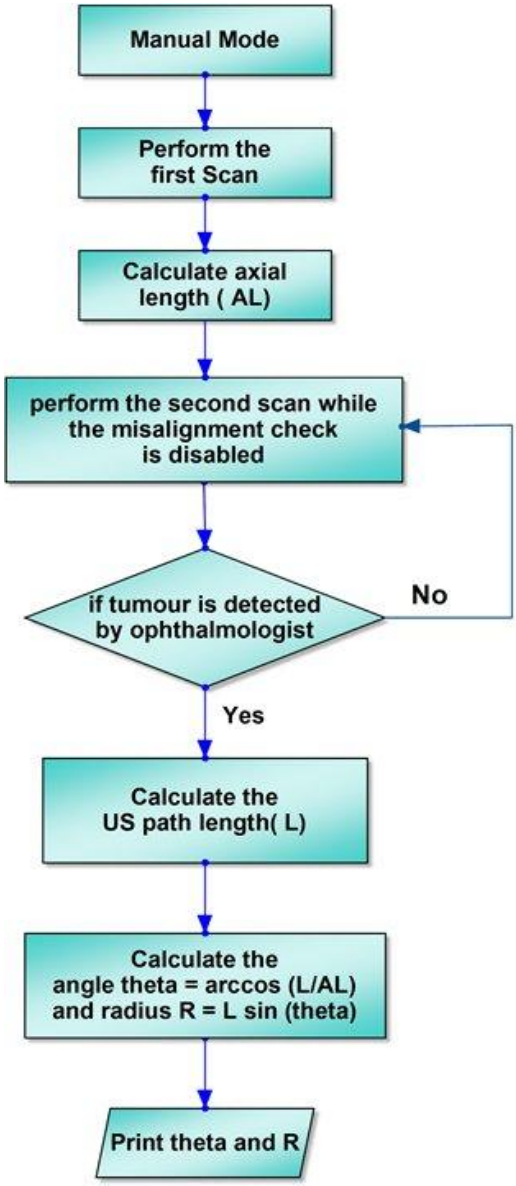


Figure 2.3-34 Non-Axial Scanning Algorithm Flowchart

2.4 References

[1] Ultrasound of the Eye and Orbit, Sandra Byrne and Ronald Green, Mosby Year Book, St. Louis, 1992 2.
[2] William R. Hendee & E. Russell Ritenour, Medical Imaging Physics, Wiley-Liss Inc., 2002

Chapter

3

Algorithm and Simulation

Introduction
DSP Tool Selection
Algorithm Block Diagram
MatLab Modules
MatLab Simulation Output

3.1 Introduction

Among the three categories of signals – analog, discrete and digital signal, A-Scans deal with a returning echo that is a pure analog signal. This signal is defined continuously in time, have an infinite range of amplitude values and can be processed using electrical devices including both active and passive elements.
Our System depends on Digital Signal Processing and the use of digital hardware to analyze, modify, condition and extract information from the signal. Thus there are two main prerequisites:

1. Converting the returning echo into a digital representation.
2. Developing an accurate algorithm for the processing and information extraction.

This chapter precisely demonstrates the sequence of operations performed by our A-Scan biometry System; that is the algorithm that will lead to the validation of all features carried out by analog and digital hardware and by software.
The algorithm is justified by the graphical and analytical results of the simulation.

3.2 DSP Tool Selection

At the algorithm development stage high level DSP tools are used to enable algorithmic level system design simulation. Some advantages of such DSP tools are:

1. Save software development time
2. Facilitate debug and modification of programs
3. Input/output operations are easy to implement and to analyze

We chose MatLab to be our DSP tool since it is an interactive, technical computing environment for scientific and engineering numerical analysis, computation and visualization. Its strength lies in:

1. Complex numerical problems can be solved easily
2. Relatively simple programming capability
3. Being easily extendable to create new functions and many other features that can be explored in the MatLab Help.

3.3 Algorithm Block Diagram

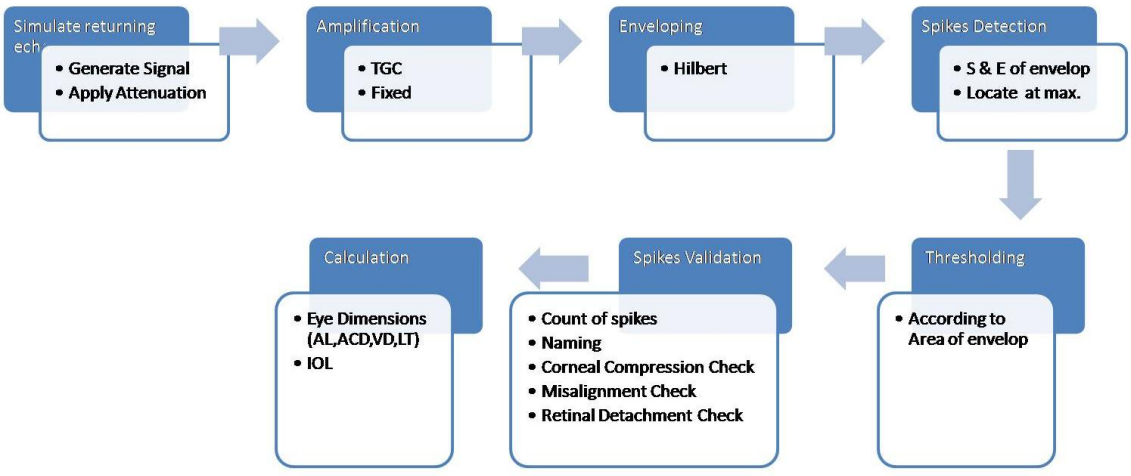


Figure 3.3-1 MatLab Algorithm Block Diagram

3.4 MatLab Modules

3.4.1 Module 1: Simulating returning echo

3.4.1.1Algorithm

Algorithm Simulate returning echo
Input(s): sampling time (t_s)
Output(s): Signal imitating returning echoes (attd)

Generate exponentially decaying sine wave with f = $1/t_s$
Create train of echoes
Generate attenuating function
Apply attenuation function on train of echoes

return (attd)

3.4.1.2 MatLab Code Excerpt

```
% ********** Generating initial signal (exp decay sin) ******************
% 4 samples per cycle and a total of 100 samples --> Sampling freq is 4MHz
t=0:samplingdelta:(2.5*10^(-6));
sig=5.*exp(-2*pi*30*10^4*t).*sin(2*pi*10*10^6*t);
% Selecting only 20 samples of signal to simulate transducer pulse
wsig=sig(1:20);

% ********** Generating 1 scan signal (multiple echos)******************
% Time axis of 884 sample for 7 echos (always multiply by the size minus 1)
t1=0:(0.025*10^(-6)):(22.075*10^(-6));
% correct scan(7 echos)
rscan=[10*wsig zeros(1,79) 3*wsig zeros(1,13) wsig zeros(1,92) wsig ...
    zeros(1,84) wsig zeros(1,404) wsig zeros(1,52) wsig zeros(1,20)];
% Noise to be added to signal for testing
noise=[zeros(1,198) 0.05*wsig zeros(1,88) 0.13*wsig zeros(1,558)];

% ********** For testing check validation ******************************
% Noise to be added to signal for testing
tamrad=[zeros(1,408) wsig zeros(1,374) wsig zeros(1,62)];
% Mixed correct & Noise signals
scanamrad=rscan+tamrad+noise;
scan=rscan+noise;

% ********** Generating attenuation function ***************************
% y=1-(1/1568)x where x is a trail of 1'sfollowed by a ramp
atten=[ones(1,99) (1-(0:1/1568:0.5))];

% ********** Multiplying attenuation signal by real one *****************
attd=scan.*atten;
subplot(611)
plot(t1,attd)
grid on
```

3.4.2 Module 2: Amplification

3.4.2.1 Algorithm

Algorithm Amplifying returning echo & compensating for attenuation
Input(s): returning echo (attd)
Output(s): amplified & TGCed signal (outsig)

Generate TGC array
Generate fixed gain array
Apply fixed gain & TGC on returning echo
Generate attenuating function
Apply attenuation function on train of echoes

return outsig

3.4.2.2 MatLab Code Excerpt

```
% ************ Amplification ***********************************************
% Generating TGC function of 4 variables with a constant region then a ramp
%
% d    _  _  _  _   ____________
%                 /|
%                /
%               /  |
%              /
% c __________/_  _|
%             |    |
%             a    b
TGC=[c*ones(1,a) (c:(d-c)/(b-a):d) d*ones(1,length(attd)-(b+1))];
% Generating Fixed gain amplifier function
fixed=[f*ones(1,length(attd))];
% Multiplying original signal by fixed gain
oppsig=attd.*fixed;
subplot(612)
plot(t1,oppsig)
grid on
% Multiplying original signal by TGC function
opsig=attd.*TGC;
subplot(613)
plot(t1,opsig)
grid on
% Multiplying original signal by BOTH TGC & fixed function & Plotting
outsig=oppsig.*TGC;
subplot(614)
plot(t1,outsig)
grid on
```

3.4.3 Module 3: Enveloping

3.4.3.1 Algorithm

Algorithm Generate Envelops for returning echoes
Input(s): amplified & TGCed signal (outsig)
Output(s): train of envelops (y)

Apply function *hilbert* on outsig

return y

3.4.3.2 MatLab Code Excerpt

```
% ********** Generating Signal Envelope **********************************
hil=hilbert(outsig);
y=abs(hil);
subplot(615)
plot(t1,y)
```

3.4.4Module 4: Spikes Detection

3.4.4.1 Algorithm

Algorithm Detect and locate spikes
Input(s): train of envelops(y)
Output(s): arrays of starts and ends of envelops(stecho,endecho)

If (difference between successive points = 10)
Start of envelop found;
Add this point to *stecho* array
If (difference between successive points = -10)
End of envelop found;
Add this point to *endecho* array
For(length of *stecho*)
Find max. value of y between start and end of each envelop
Locate start of envelop at max. value

return stecho,endecho

3.4.4.2 MatLab Code Excerpt

```
% ********** Lead Edge Detection Module **********************************
% A loop to detect starting and ending points of envelopes
while ok
    if ((y(m)< 10) & (y(m+1) > 10))
        stecho = [stecho (m+1)];
    end;
    if((y(m)>10) & (y(m+1)<10))
        endecho=[endecho m];
    end;
    m=m+1;
    if(m==length(hil))
        ok=0;
    end;
end;
```

```
% ********** Spike detection
*****************************************
for n=1:length(stecho)
mx=max(y(stecho(n):endecho(n))); % find max value of entire array
[r,c]=find(y==mx);  % find max position
stecho(n)=c(1);
end
```

3.4.5 Module 5: Thresholding

3.4.5.1Algorithm

Algorithm Thresholding
Input(s): arrays of starts and ends of envelops(stecho,endecho)
Output(s): array of spikes (thresh)

Calculate area of envelops using function *trapz*
Create array with values of areas
If (area>= threshold)
Keep it in array *thresh*
Else
Set to zero
Create array *final* consisting of all kept spikes

return thresh

3.4.5.2MatLab Code Excerpt

```
% ********** Thresholding  **********************************************
% Calculating area under envelopes by Trapz function
allpulses=[];
for n=1:length(stecho)
allpulses=[allpulses trapz(y(stecho(n):endecho(n)))];
end;
thresh=[];
time=[];
% A loop to extract pulses above thresholds and there LOCATIONS in the
% array ("time")
for n=1:length(allpulses)
    if(allpulses(n)>=thr)
% Putting above threshold peaks in new array
            thresh=[thresh allpulses(n)];
            time=[time stecho(n)];
    else
% Resestting value of below threshold peak in original array
        allpulses(n)=0;
        end;
end;
% Generating Correct thresholded signal in final array
final=[];
for n=1:length(allpulses)
    if n~=length(allpulses)
% Concatinates the final array with zeros(between each peak) and the
% peak themselves
    final=[final allpulses(n) zeros(1,stecho(n+1)-1-stecho(n))] ;
    else
```

```
% Special concatination to final array as no more peaks are
% available at the end of the signal
     final=[final allpulses(n) zeros(1,length(y)-stecho(n))];
    end;
end;
subplot(616)
plot(t1(1:length(final)),final)
grid on
```

3.4.6 Module 6: Spikes Validation

3.4.6.1 Algorithm

Algorithm Validation
Input(s): array of spikes (thresh)
Output(s): names of spikes, results of corneal compression and misalignment check

Label each spike with its name using function *naming*
Calculate thickness of cornea
If(< pre-defined thickness)
Give alarm and reset scan
Check misalignment using functions *checklensper* &*checkretper*

return void

3.4.6.2 MatLab Code Excerpt [1]

```
% *********** Check Validation Module ************************************
% Determining number of above threshold pulses
pulseno=length(thresh);
% For correct no of spikes
if(pulseno==actualpno)
% Calculating Corneal thickness using samples, sampling time & US
% speed in cornea
        corneathickness=((time(3)-time(2)-
echowidth)*samplingdelta*USinCornea);
% Generating the names for the pulses based on case
        names=naming(pulseno,actualpno,actualtime,actualnames,time);
% check corneal compression
        if(corneathickness<realCdistance)
% Call Alarm & Reset
        else
% check for misalignments
          p1 = checklensper(actualpno,thresh);
          p2 = checkretper(pulseno,thresh,actualpno,names);
.....
.........
```

1 The used functions are found in Appendix III.

3.4.7 Module 7: Calculations

3.4.7.1Algorithm

Algorithm Calculate
Input(s): array of spikes (thresh)
Output(s): Eye Dimensions (AL, ACD, VD, TL) & IOL

Calculate AL, ACD, VD, TL using function *calculate*
Calculate IOL using most suitable formula with function *auto*

return AL, ACD, VD, TL & IOL

3.2.7.2MatLab Code Excerpt [1]

```
% ********** Calculations ***********************************************
[AL,ACD,VD,LT]=calculation(alltime,allintensity,2,pulseno-
1,porpseudo,typeorthick,value,trauma)%we are sending no =1 for testing
% IOL calculation
                            if(aIOL==1)
                                IOL= auto(AL,K1,K2,Ref)
                            end;
                        end;

                    else
                    acceptedscan=0
                    end;
                end;
            end;
        else
% call alarm & reset
.....
.........
```

1 The used functions are found in Appendix III.

3.5 MatLab Simulation Output

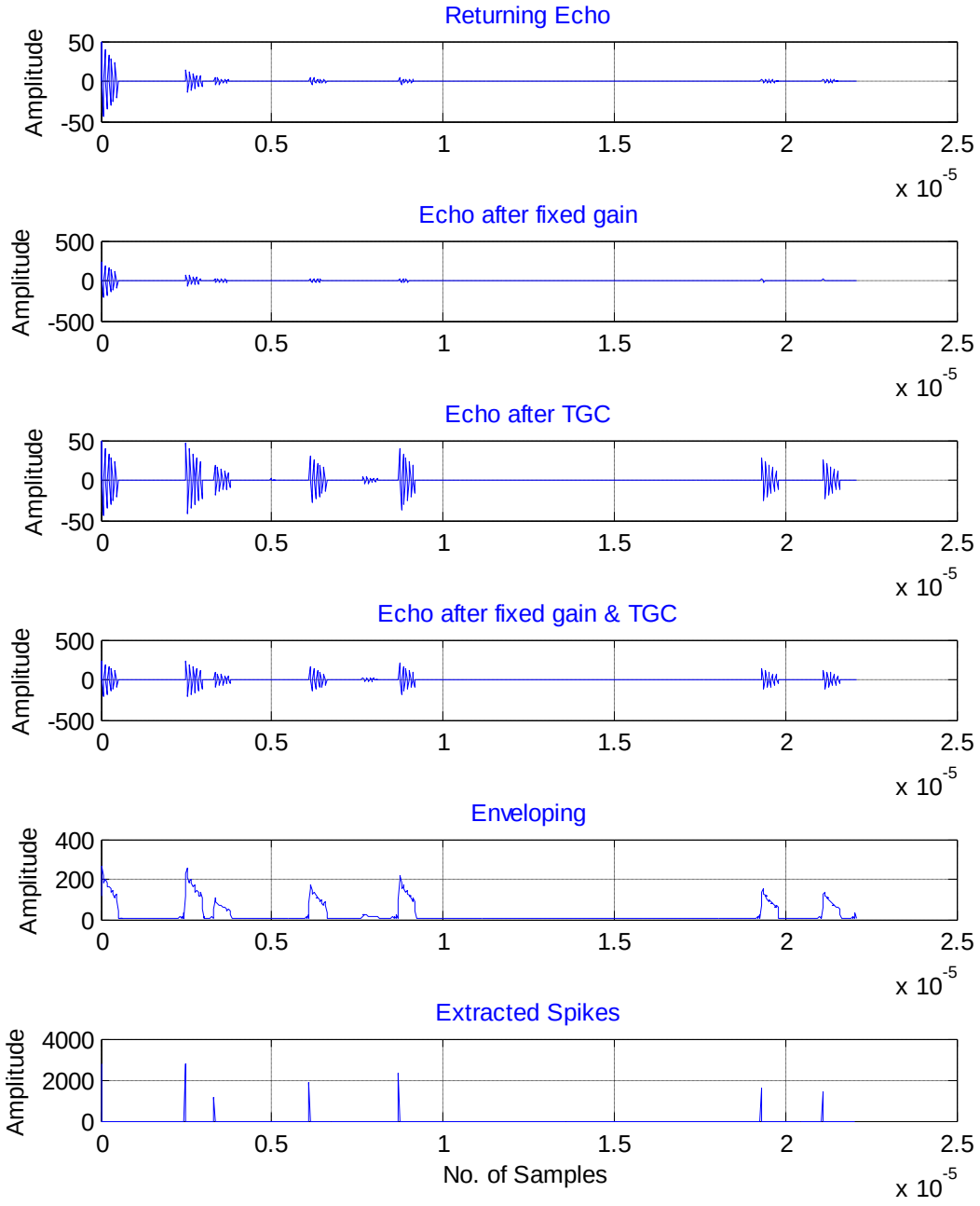


Figure 3.5-1 MatLab Siumlation Signal Output

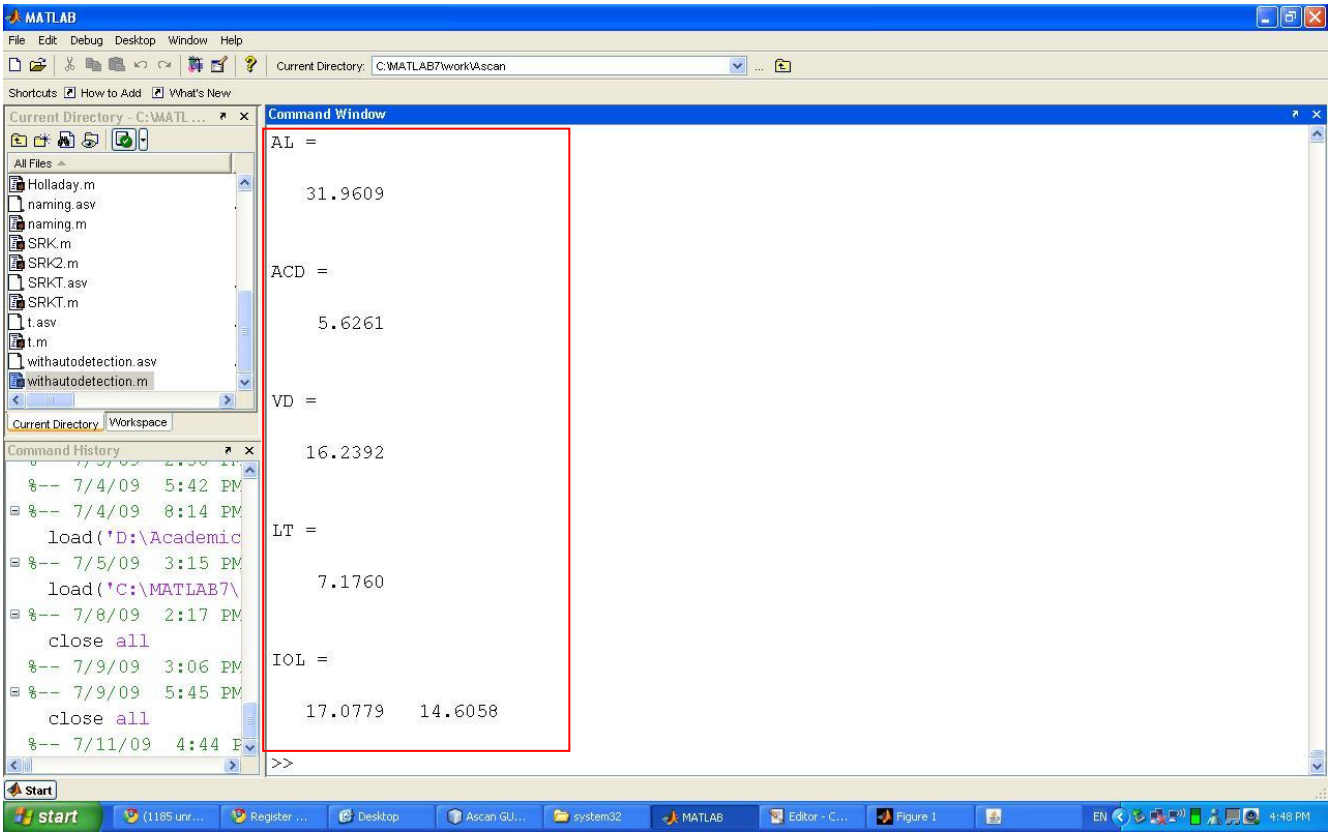


Figure 3.5-2 Snapshot of MatLab Command Window. AL, ACD, VD, TL and IOL are calculated

Chapter 4

Future Work

Introduction	**Resulting Model**
Signal Processing Hardware	**Data and Values Collection**
Software	**TGC Calculation**
Interface	**References**

4.1 Introduction

When creating a complete A-Scan Biometry system, it is important to consider a wide variety of interacting factors that contribute to the overall efficiency, performance and cost of the system. In order to better understand these considerations we will identify the key characteristics of the main subsystems Signal Processing Hardware, Software, and Interface and identify how each sub-subsystem interacts and also influences the design.
Preparing for the implementation step the device selection process plays a decisive and critical role since all subsequent phases will build on it. Its success depends on the accuracy of the collected data. Thus data aggregation and collection is of great necessity and guarantees compatibly of each sub-system with the data it works on. Appendix V contains all gathered data and values essential fo designing e-DIOLA scan. This Chapter documents the device selection procedure as well as the resulting model of the sub-system due to this selection.

4.2 Signal Processing Hardware

The Processing of the signal is carried out using DSP hardware. Although it is possible to implement DSP algorithms on any digital computer, the throughput (processing rate) determines the optimum hardware platform. Four DSP platforms are widely used for DSP applications [1]:

1. General purpose microprocessors and microcontrollers (µP and µC)
2. General purpose digital signal processors (DSP chips)
3. Digital building blocks (DBB) such as multiplier, adder, program controller and
4. Special purpose devices such as application specific integrated circuits (ASIC)

The characteristics of the four platforms are compared in Table 4.1.

	ASIC	**DBB**	**µP and µC**	**DSP chip**
Chip count	1	>1	1	1
Flexibility	None	limited	programmable	programmable
Design time	long	mdeium	short	short
Power consumption	low	Medium-high	medium	Medium-high
Processing speed	High	high	Low-medium	Mediu-high
Reliability	high	Low-medium	high	High
Development cost	high	medium	low	Low
Production cost	low	high	Low-medium	Low-mdeium

Table 4.2- 1 Comparison of different Hardware platforms

A proper choice from the many available characteristics requires a full understanding of the processing requirements of the DSP system under design. The objective is to select the device that meets the system's time-scale and provides most cost-effective solution according to certain criteria.

After intensive investigations and search we came to the decision of using the C8051F020 microcontroller developer kit manufactured by Silicon Laboratories. It satisfied the above criteria as follows:

Criteria	Required	Provided
Efficient data flow	Real time data transmission	YES
Computational power	Medium	High
Resolution	Medium (8bits)	High (up to 12 bits)
Processing speed	High	High
Memory size	Medium	Medium
Signal type	Mixed signal (Analog and digital)	Mixed signal
Availability of a full set of development tools and supports	C compiler, linker, logic analyzer, development assistance, real time debugging tool	All available with the kit

Table 4.2- 2 Criteria of Selection

4.3 Software

Among the many available and emerging Software we had to make the right choice concerning the programming language that will be used in e-DIOLA scan product. Comparing Java of Sun Microsystems with its counterparts, Java showed the following advantages [2]:

- **Java is simple:** Java was designed to be easy to use and is therefore easy to write, compile, debug, and learn than other programming languages. The reason that why Java is much simpler than C++ is because Java uses automatic memory allocation and garbage collection where else C++ requires the programmer to allocate memory and to collect garbage.
- **Java is object-oriented:** Java is object-oriented because programming in Java is centered on creating objects, manipulating objects, and making objects work together. This allows you to create modular programs and reusable code.
- **Java is platform-independent:** One of the most significant advantages of Java is its ability to move easily from one computer system to another.
- **Java is distributed:** Distributed computing involves several computers on a network working together. Java is designed to make distributed computing easy with the networking capability that is inherently integrated into it.
- **Java is interpreted:** An interpreter is needed in order to run Java programs. The programs are compiled into Java Virtual Machine code called bytecode.
- **Java is secure:** Java is one of the first programming languages to consider security as part of its design. The Java language, compiler, interpreter, and runtime environment were each developed with security in mind.
- **Java is robust:** Robust means reliable and no programming language can really assure reliability. Java puts a lot of emphasis on early checking for possible errors, as Java compilers are able to detect many problems that would first show up during execution time in other languages.
- **Java is multithreaded:** Multithreaded is the capability for a program to perform several tasks simultaneously within a program. In Java, multithreaded programming has been smoothly integrated into it, while in other languages, operating system-specific procedures have to be called in order to enable multithreading.

Hence Java has certainly proved itself the most suitable software platform not only by the numerous advantages it provides but also with its user friendly Netbeans IDE and open source libraries and developing tools.

4.4 Interface

Having settled on the hardware and software platforms we had to choose a proper communication channel to ensure reliable and correct data transfer. Commonly used parallel ports and/or serial ports showed a remarkable number of shortcomings:

- Interrupts
- I/O addresses
- Cabling
- Non-sharable interface
- No hot attachment
- Installation

While the universal serial bus (USB) interface and communication protocol had the following advantages:

- Fast
- Bi-directional
- Isochronous
- low-cost
- dynamically attachable serial interface
- Backward compatible
- High-performance (up to 480Mbps for USB2.0 and 12Mbps for USB1.1)
- Provides automatic error detection and recovery
- Provides cable power and power management features
- consistent with the requirements of the PC platform of today and tomorrow
- Completely external to the PC
- Simple and standard connector

Clearly, it would have been unwise to use anything but a USB interface since it fulfills all necessary prerequisites that grant a fast, reliable and error free data transmission. But this would not happen until matching of communication nature is achieved. Thus we need to convert the data sent serially by the MCU's UART to USB in order to be accepted by the USB port and consequently the Java Platform.
Just as famous as the USB interface was the FTDI Serial ↔USB data transfer chip (FT232BQ) that features:

- full handshaking signals
- different data package definitions
- selectable baud rates
- USB bulk or isochronous transfer modes
- USB 1.0 and USB 2.0 compatible

- In board programmable external EEPROM
- Integrated 3.3V regulator for USB I/O
- Virtual COM and D2XX supporting Drivers for divers Platforms

More features and details can be seen by following the URL to the datasheet in Appendix VIII.
This way we have selected the components of our system and assured compatibility and harmony among them. It is now time to select the most suitable model for the design and this is done in the next section.

4.5 Resulting Model

Knowing the nature and type of each component it is required to divide roles in a manner that guarantees efficiency and performance of e-DIOLA scan. According to the Market survey results demonstrated in section 1.5 it was clear that the stand alone model is dominant among all A-scan biometry systems while the PC-based and Laptop –USB –based models were less frequent to occur. Consequently we brainstormed and debated the pros and cons of each model.

	Standalone system	**Microcontroller - PC based**	**Software**
Advantages	1-No compatibility issue	1-Friendly UI	1- Friendly UI
	2-Independent on electrical problems	2-Independent processing	2-Faster processing
	3- No interfacing problems	3-Extended data storage	3-Extended data storage
	4-More acquired knowledge(e.g.: power supply)	4-More compact	4-More features
	5-More acceptable	5-More Features by software	5-More compact (probe + CD)
	6-Needs little training	6-Portable(battery)	6-Inexpensive
	7-Inexpensive		
Disadvantages	1-Difficult to implement	1-Dependent on PC or laptop	1-Specific requirements of processor
	2-Limited data storage	2-Restricted by complex multitasking	2-Higher risk of computer freezing or limitation
	3-Not friendly software	3-Compatibility issues	3-Compatibility issues
	4-Limited processing	4-Dependent on laptop power supply	4-Upgrading
	5-Integrated printer (so more complex)	5-Specific requirements of laptop or PC	5-Dependent on laptop power supply
		6-Viruses' problem (no windows network)	
Votes	**28.3%**	**51.7%**	**20%**

Table 4.5 -1 Votes for Pros and Cons of different Systems

Since the votes were in favour of the microcontroller-PC-based model it was this hybrid model that we employed in designing e-DIOLA scan. The hybrid nature is demonstrated in detail in Part II were the system design is put in plain words.

4.6 Data and Values Collection

In order to produce a sophisticated software, we needed to determine the precise values of: thickness of each interface within the eye, specific ultrasound velocities, minimum and maximum length of the human eye.
We have managed to collect the required values from various sources as ultrasound books, eye anatomy and ophthalmologists [1],[2]. The concluded values are shown in the table below, you may view **Appendix V** for more details.

Eye interface	Thickness (mm)	Velocity (m/s)
1. Cornea	0.52	1620
2. Aqueous	3.16	1500
3. Lens	4.3 (min:2.62 , max:6)	Normal lens :1620 or 1641
		Silicon : 1000
		PMMA: 2760
		Acrylic: 2120
4. Vitreous	19.5 (min:18.23 , max:21.99)	1520 or 1532
5. Retina	1.85	1540
6. Sclera	0.55 (min:0.4 , max:0.6)	1613 -1622

Table 4.6 -1 Summarized Table of values

The human eye has a minimum axial length of (21 mm) and a maximum of (29.88mm) giving an average of 24.05mm. Thus, the maximum length to be measured by the A-probe is 29.88 mm.

4.7 Time Gain Compensation (TGC) Calculation

It is known that the ultrasound signal strength is attenuated by0.1 dB/cm using 1MHZ [3],[4].
By applying this fact on our system frequency which is 9MHz, the attenuation coefficients of the ultrasound signal in the different interfaces of the eye were calculated as shown in the table below:

Interface	Cornea	Aqueous	Lens	Vitreous	Retina-Sclera
Attenuation coefficient (dB/mm) at 9MHz	0.234	0.091	1.82	0.091	0.364

Table 4.7 – 1 attenuation values for interfaces of the human eye

To know how much the signal is attenuated, we need to obtain the distance between the spike of each interface and the spike of the next interface. But, in order to calculate the distance, the times need to be calculated first.

Using the indices acquired from the microcode and dividing by the sampling delta which is 4.55MHz, the time was calculated as shown in the table below:

Peak	Anterior Cornea	Posterior Cornea	Anterior Lens	Posterior Lens	Retina	Sclera
Max. Index	0	1	14	27	128	139
Time	0	1.0989E-7	1.5384E-6	2.96703E-6	1.4065E-5	1.527E-5
Min. Index	0	5	33	66	198	210
Time	0	5.4945E-7	3.62637E-6	7.25274E-6	2.1758E-5	2.307E-5

Table 4.7-2 Time of echo returning from each interface

Thus, we were able to calculate the distances as shown in the table below:

Region	Cornea	Aqueous	Lens	Vitreous	Retina-Sclera
Min. distance (mm)	0.17802198	2.142857	2.3285713	16.925825	1.8615385
Max. distance(mm)	0.89010984	4.6153846	5.9109893	22.12088	2.03077

Table 4.7 - 3 Distance of each interface

Multiplying the attenuation Coefficients (dB/mm) given in Table 4.7 -1 and the max. distance for each interface of the eye given in Table 4.7 -3 we can calculate the attenuation in (dB) as shown in the table below:

Region	Cornea	Aqueous	Lens	Vitreous	Retina-Sclera
Attenuation at 9MHz	0.234	0.091	1.82	0.091	0.364
Max Distance (mm)	0.8901098	4.6153846	5.9109893	22.12088	2.03077
Attenuation (dB)	0.208285	0.419999	10.75800	2.013000	0.739200

Table 4.7 - 4 Attenuation of each interface in dB

Now, we have the attenuation for each interface separately, but we need to know by how much the received ultrasound signal will be amplified, so we must calculate the amplification value for each received echo using the rule:

$$dB = 20 \log A/A0$$

There are 5 echoes returning from the eye: Posterior Cornea echo, Anterior Lens echo, Posterior Lens echo, Retinal echo, Sclerlal echo. The amplification value is calculated as shown in the table below:

Region	Echo 1	Echo 2	Echo 3	Echo 4	Echo 5
Attenuation (dB)	0.208285	0.628284	11.38628	13.399284	14.138484
Attenuation Multiplied by 2	0.41657	1.256568	22.77256	26.79856	28.276968
Amplification Value	1.049	1.1556	13.76	21.873	25.93

Table 4.7- 5 Amplification value for each region

So, we can conclude that the max. amplification value is 26, but according to the Fixed Gain Amplification module the amplification value is 10.
Since 26/10=2.6 ~3
So, we can reach the suitable values for the Time Gain Compensation parameters:
a=0
b=215 → as the sclera spike is the furthermost one
Lower gain → c=1
Higher gain → d=3

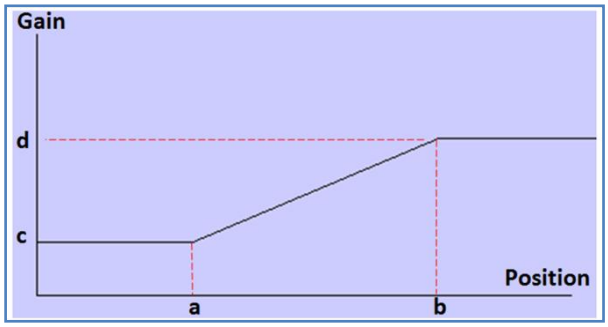


Figure 4.7 - 0-1 Final TGC curve

4.8 References

[1] Wolf's Anatomy
[2] Real Time Ophthalmic Ultrasonography and Biometry, Richard Koplin, Martin Gersten, and Barton Hodes, SLACK Inc., 1985
[3] Medical Imaging Physics, William R. Hendee, E. Russel Ritenour, fourth edition, 2002
[4] Ultrasonic Bioinstrumentation, Douglas A. Chrostensen, University of Utah, 1998

Part II

System Design

Chapter 5

Hardware

Introduction	**Enveloping Circuit**
Transducer	**Bandpass filter**
Pulser Circuit	**References**
T/R switch	

5.1 Introduction

Carrying out certain parallel processes leads to realization of the design. Figure 1.6-1 showed the branching into software and hardware paths. This branching is demonstrated in this Part of the document and is further detailed into firmware and interfacing in addition to the software and hardware branches.
This chapter will focus on the hardware module (Figure 5.1), illustrating the schematic circuit and function of each operational block and verifying it with test-snapshots.

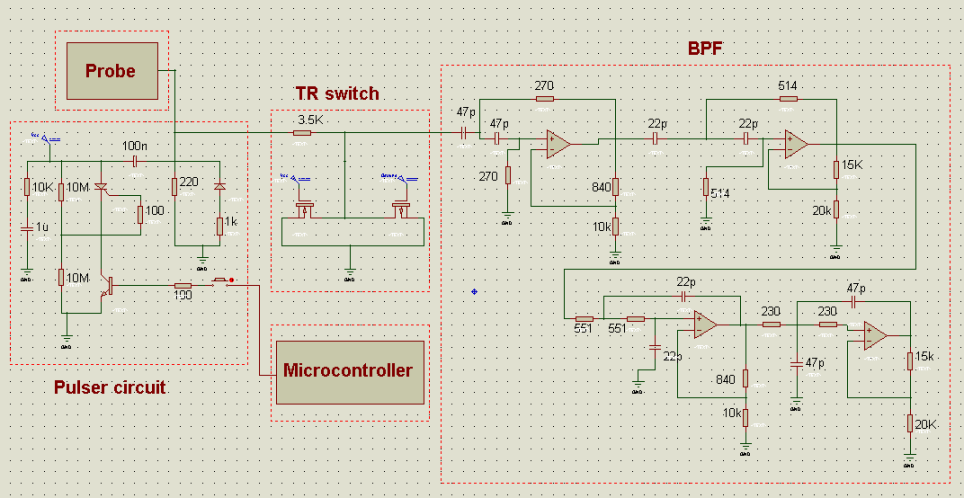


Figure 5.1-1 Overall Block Diagram of hardware module

5.2 Transducer

Schematic:

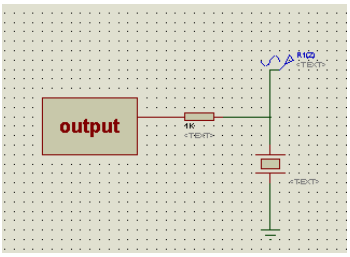


Figure 5.2-1 Transducer Schematic

Picture:

Figure 5.2-2 Transducer

Function: A typical A-scan biometry probe is a single piezoelectric crystal, the size of a pen that produces ultrasound waves when a voltage is applied trough the piezoelectric effect

Verification:

To determine the resonance frequency, the following steps were performed:

1. 1K resistor connected in series with the ultrasound crystal
2. Applied a voltage on the crystal, and measured the output voltage on the other terminal of the resistor.
3. While changing the frequency of the applied voltage waveform, we noticed variation of the output amplitude.
4. The amplitude was maximum at a frequency of 9MHz (resonance) as shown on the digital oscilloscope.

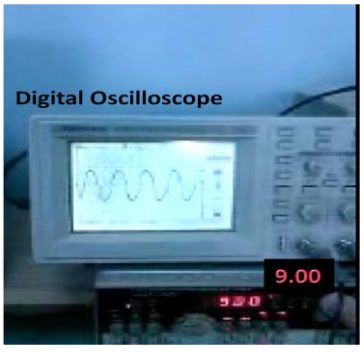


Figure 5.2-3 Transducer Test Result

5.3 Pulser Circuit

Schematic:

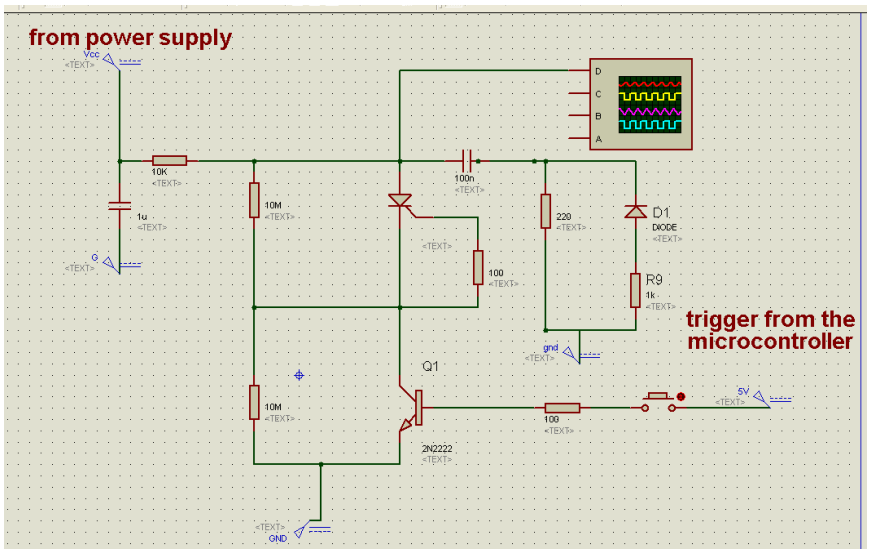
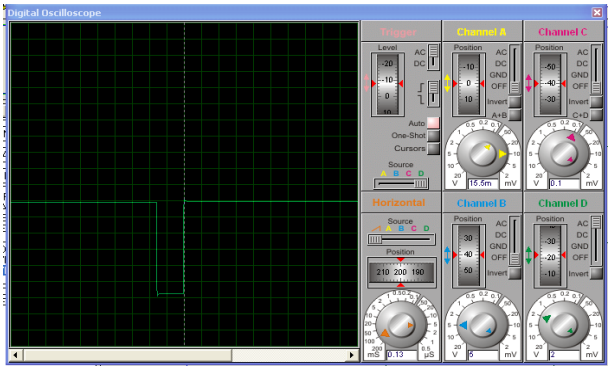


Figure 5.3-1 Pulser circuit schematic

Function: The pulser circuit energizes the transducer of the US probe by a relatively high voltage–short duration pulse to produce ultrasound waves.
Initially, the 25 supply volts are divided equally across the resistors R_1 and R_2, as R_3 is relatively small and can be ignored. When the circuit is triggered from the microcontroller:
The transistor Q is turned ON → R_2 is short circuit.
Now, there is V_s across the SCR → causing forward break-over.
While SCR and transistor Q are conducting, capacitor C_1 discharges through this series circuit. Thus, the voltage at point X falls rapidly from V_s to the ground.
As the voltage across the SCR is decreased, it will switch back to the blocking state and a positive voltage across X begins.

Verification: As shown below, a voltage pulse is produced when the microcontroller triggers the circuit.



5.4 T/R switch

Schematic:

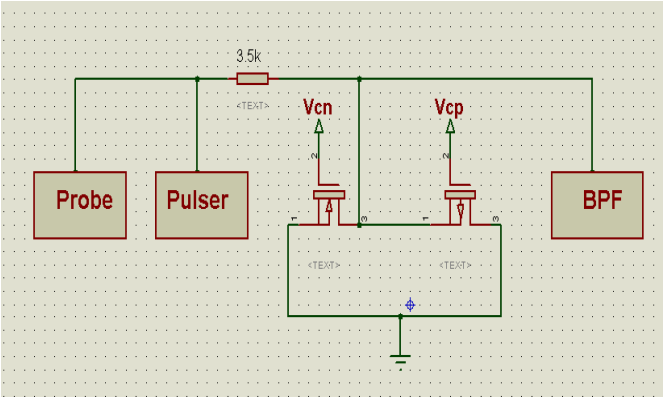
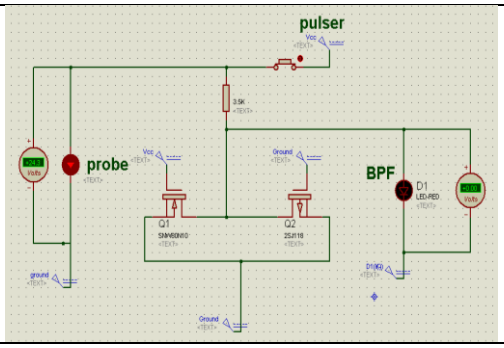
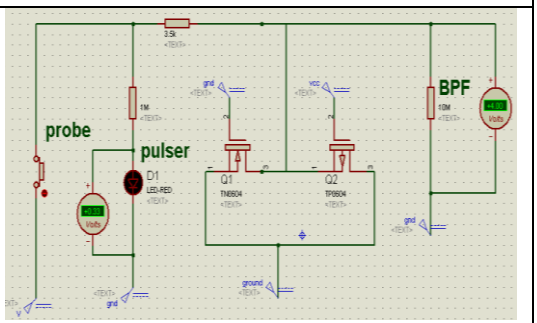


Figure 5.4-1 T/R switch schematic

Function: The Transmitter/Receiver switch is provided to prevent the large signals from the pulser circuit from finding their way into the band pass filter.
When the V_{cn} is *high* and V_{cp} is *low*: the pulser output signal finds its way to trigger the probe while the TR switch prevents it from reaching the band pass filter.
When the V_{cn} is *low* and V_{cp} is *high*: the returned signals from the tissue through the probe finds their way to the band pass filter while the TR switch prevents them from reaching the pulser circuit.

Verification:

	V_{cn}	V_{cp}	Pulser	BPF	Probe	Simulation
Case 1	high	low	operating	0 V	pulse from the Pulser circuit.	
Case 2	low	high	Zero input signal	signal from the probe	receives signal from tissue	

5.5 Enveloping

Schematic:

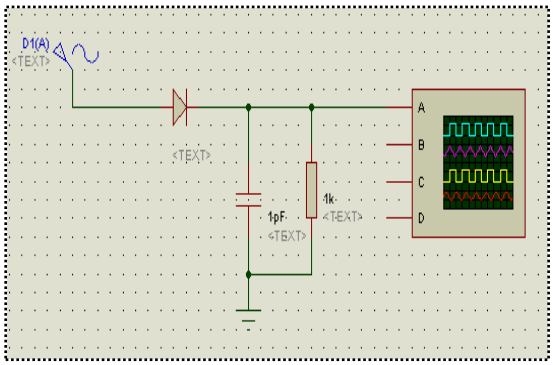
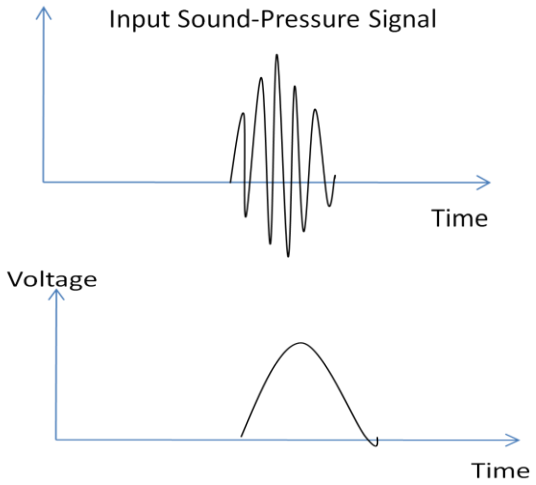


Figure 5.5-1 Envelope Generation Schematic

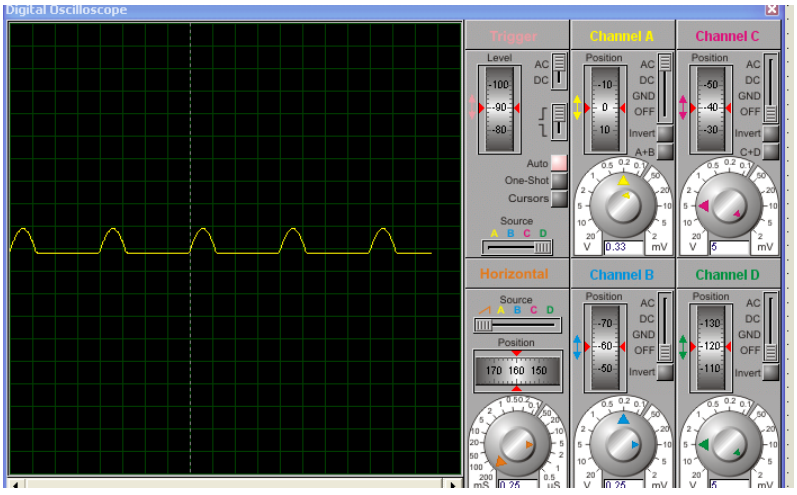
Picture:



Envelope-Detector Response to a realistic A-Scan Pulse

Function: "Enveloping" is another term for "low pass filtering". When a complex signal is passed through a low pass filter, the resulting signal is the envelope (outline) of the original

Verification: upon applying a sinusoidal input the rectified output was shown on the oscilloscope



5.6 Bandpass Filter

Function:

Before encountering the ADC, the signal is processed by an anti-alias filter to prevent aliasing during sampling. We used a technique known as *bandpass sampling*, to sample a continuous bandpass signal that is centered about a non-zero center frequency. Utilizing a modified sallen-key circuit, we designed and implemented a four-pole-2-stage band pass filter of Bessel type for:

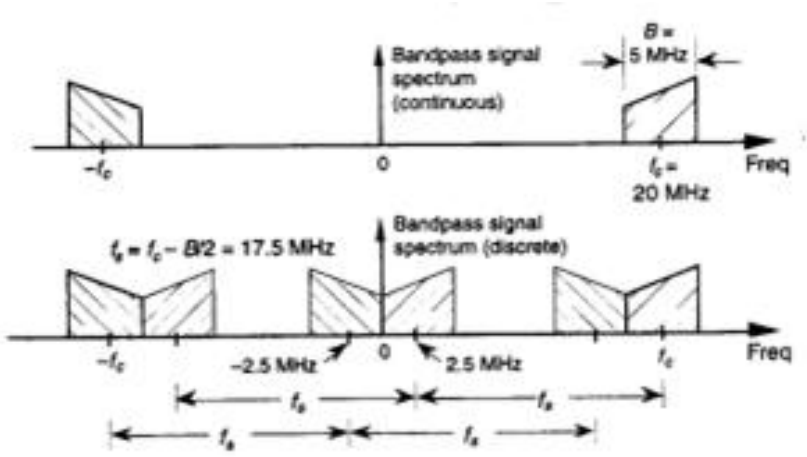
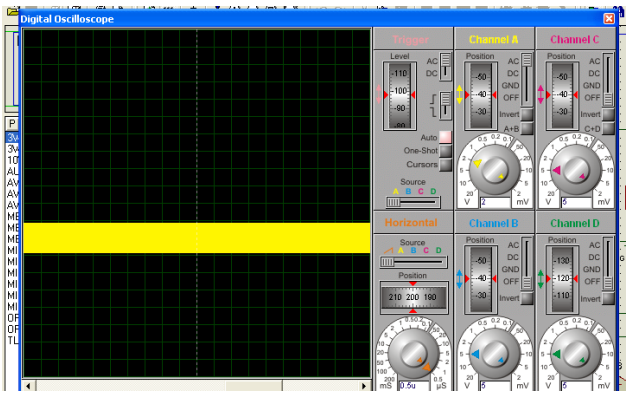
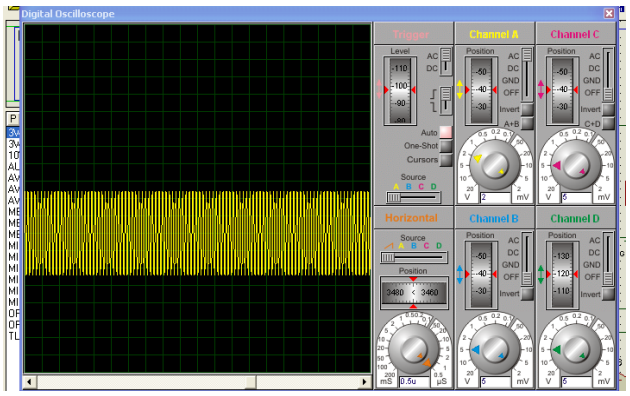
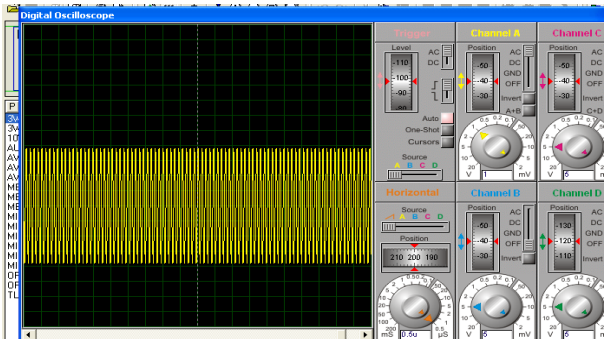
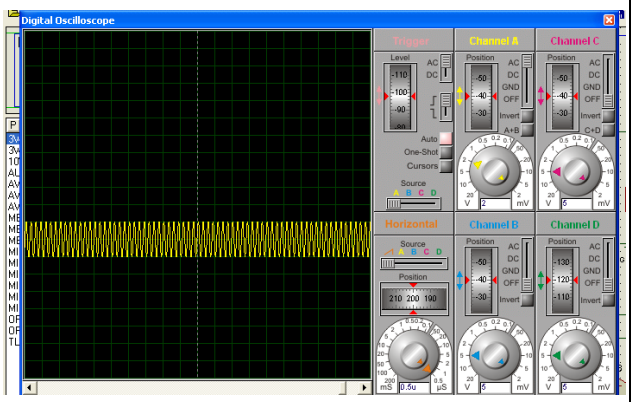
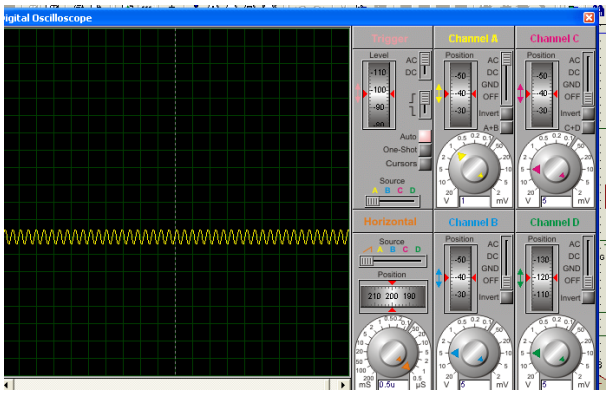


Figure 5.6-1 Band pass filter characteristics

- Sharper cutoff frequencies.
- Removing ripples in the pass band.
- Eliminating overshooting and ringing problems. [3],[4]

Verification: The table below shows the output voltage amplitude of the BPF relative to the input amplitude at particular input frequencies.

Frequency (MHz)	Output	Output Amplitude (V)
25		2 V
9.6 MHz		5 V

	8.76 MHz		4 V	
	7 MHz		0.2 of the input amplitude	
	4.38 MHz		0.5V	

5.6.1 Mathematical Derivation

Bandpass sampling not only reduces the speed required for the ADC, but it also reduces the amount of digital memory necessary to capture a given time interval of a continuous signal. In bandpass sampling, we are more concerned with signal bandwidth than with its highest frequency components. Our bandpass signal's highest frequency is 9MHz. Conforming to the Nyquist criterion implies that the sampling frequency be 18MHz to avoid aliasing, provided that all band signal contains information.

In our case, information is band limited around 9MHz, so we won't need all 18MHz to sample the signal. Thus, the efficient sampling frequency is calculated using the law below:

$$\frac{2fc - B}{m} \geq fs \geq \frac{2fc + B}{m + 1}$$

Where

fc is corner frequency which is 9MHz

B is bandwidth of signal assume it to be 0.2MHz

m is number of replications which is a positive number and integer

m	$\frac{2fc-B}{m}$	$\frac{2fc+B}{m+1}$	Optimum sampling rate
1	17.8	9.1	
2	8.9	6.07	
3	5.93	4.55	
4	4.45	3.64	
5	3.56	3.03	
6	2.967	2.6	
7	2.54	2.275	
8	2.225	2.022	
9	1.978	1.82	
10	1.78	1.654	
11	1.618	1.516	
12	1.48	1.4	
13	1.369	1.3	
14	1.27	1.21	
15	1.187	1.138	
16	1.11	1.07	
17	1.047	1.01	
18	0.99	0.96	
19	0.937	0.91	
20	0.89	0.87	
21	0.85	0.83	
22	0.81	0.79	
23	0.77	0.76	
24	0.74	0.73	
25	0.71	0.7	
26	0.68	0.67	
27	0.66	0.65	
28	0.64	0.63	0.64
29	0.61	0.61	

Table 5.6 – 1 Band pass filter coefficients

It is clear from Table 5.1 above, that two regions are equal as at m=29. Thus, we selected 0.63 to be the system's sampling frequency. For safety measures, we raised the sampling frequency to 0.64MHz. Starting from 0 MHz, the HPF frequency is 8.76MHz and LPF frequency is 9.16MHz.

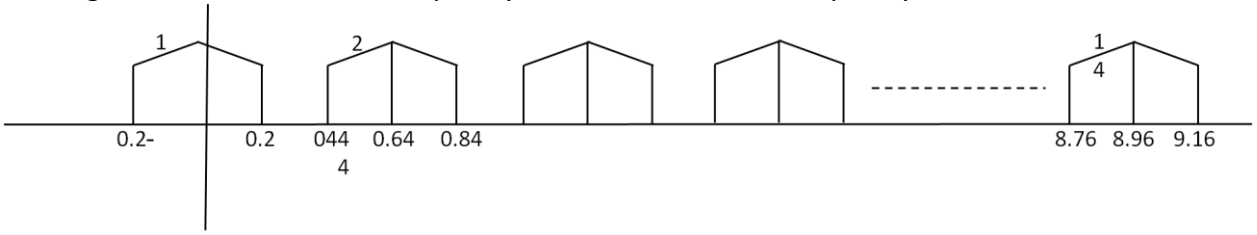


Figure 5.6-2 Illustration of **sampling** frequency effect

5.6.2 Bandpass filter design

For more accurate measurements we choose four order filtering [1]. Once selecting Bessel as our BPF type, particular parameter values (K1, K_2) are set according to the number of poles used. By using these set values of (K_1, K_2) , we assigned the most suitable values for R_1 ,R_f and C that best fit our corner frequency f_c by utilizing the relations below:

$$R = \frac{K1}{Cfc}$$

$$Rf = R1\ K2$$

Stage	K1	K2
1	0.1111	0.084
2	0.0991	0.759

Stage	HPF	LPF
1	C=47Pf R=270$\boldsymbol{\Omega}$ Rf=840$\boldsymbol{\Omega}$ R1=10K$\boldsymbol{\Omega}$	C=22Pf R=551$\boldsymbol{\Omega}$ Rf=840$\boldsymbol{\Omega}$ R1=10K$\boldsymbol{\Omega}$
2	C=22Pf R=514$\boldsymbol{\Omega}$ Rf=15K$\boldsymbol{\Omega}$ R1=20K$\boldsymbol{\Omega}$	C=47Pf R=230$\boldsymbol{\Omega}$ Rf=15K$\boldsymbol{\Omega}$ R1=20K$\boldsymbol{\Omega}$

5.7 System Power Supply

Schematic:

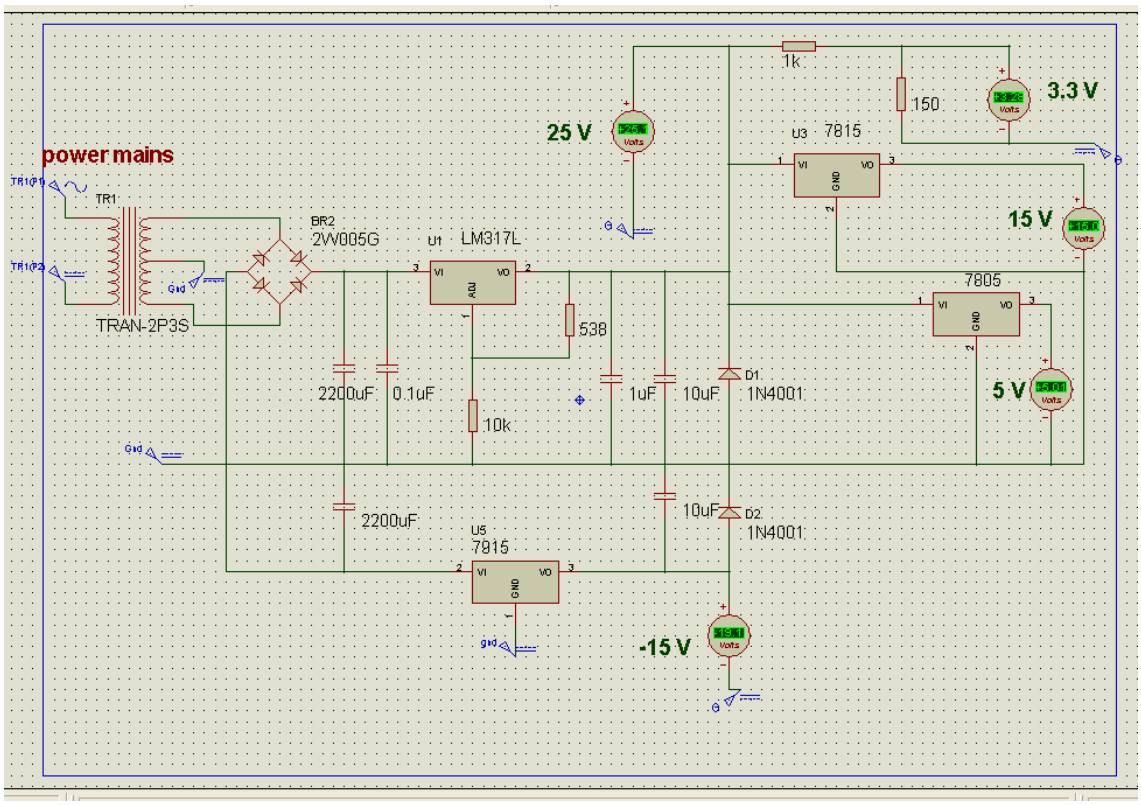


Figure 5.7-1 Power Supply Schematic

Function: Supply the system with 5V, ±15V, 3.3V and 25V.

Verification:

Required	Obtained	Required	Obtained
1. 25 V		**2. 15 V**	
3. - 15 V		**4. 5 V**	

5. 3.3 V	

5.7.1 Power Supply Design

In the main-supplied electronic system, the AC input voltage V_{AC} must be converted into a DC voltage with the right value and degree of stabilization. This is achieved using a rectifier circuit in a bridge configuration Figure (5.7-1).
In this basic configuration, the peak voltage across the load is equal to the peak value of the AC voltage supplied by the transformer's secondary winding. Also, we found out that the output voltage waveform is improved considerably when a filter capacitor is added after the rectifier diodes. We used a transformer of (28-0-28) V as maximum voltage to produce 25 V.

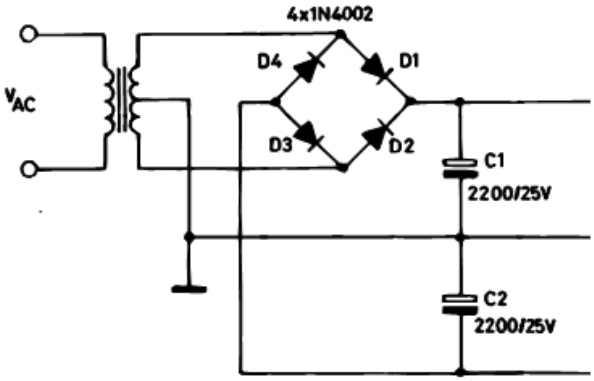


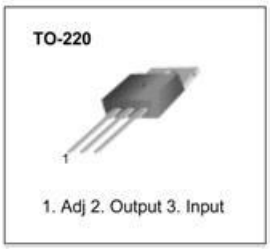
Figure 5.7-2 Rectifier Circuit -Bridge Configuration

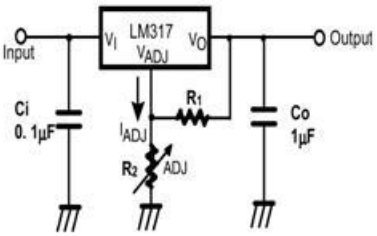
Requirements:
After we designed all hardware modules, we concluded the following desired voltage values:

Hardware Module		**Desired Voltage (V)**
1.	FT232	5
2.	Micro kit	3.2
3.	BPF	± 15
4.	pulsar circuit	25

To provide the above required voltages, we used different regulators; all are mentioned below. For further details refer to Appendix VI.

25 volt





- **Component:** LM317 variable positive voltage regulator
- **Input:** At pin 3→ 28 volts
- **Output:** At pin 2→ 25 volts
- **Calculation:**

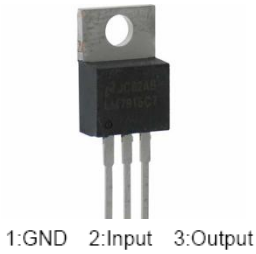
$$V_O = 1.25V\ (1 + R_2 / R_1) + I_{ADJ}\ R_2$$

Since I_{ADJ} is controlled by less than 100µA we assumed it to be 50 µA. If we set that R2=10KΩ. By substitution we get that R2=538Ω

+ 15 volt

- **Component:** LM-7815 positive voltage regulator
- **Input:** At pin 1→ 25 volts
- **Output:** At pin 3→ 15 volts
- **Calculations:** no need for calculation

-15 volt



- **Component:** LM-7915 negative voltage regulator
- **Input:** At pin 2→ -25 volts
- **Output:** At pin 3→ -15 volts
- **Calculations:** no need for calculation

5 volt

- **Component:** LM-7805 positive voltage regulator
- **Input:** At pin 1→ 25 volts
- **Output:** At pin 3→ 5 volts
- **Calculations:** no need for calculation

3.2 volt

- Our micro kit operate on range of voltage from 2.7-3.6 volt
- Maximum current that kit Borne is 800mA
- Analog supply current is 1.7mA
- Digital supply current is 10mA
- For safety we use 20mA

We didn't find a regulator that yields 3.2 V, so we had to design one to function as desired. The design consists of two resistors to apply voltage division across them:

- Input: 25 volt
- Output: 3.2 volt
- Calculations: $\mathrm{Vout} = \frac{\mathrm{R2}}{\mathrm{R1}+\mathrm{R2}}\ \mathrm{Vin}$
- R2=150$\boldsymbol{\Omega}$
- R1=1K$\boldsymbol{\Omega}$

5.8 Printed Circuit Board

We designed the PCBs using **OrCAD 10.5** compatible with windows XP
We have undergone the following steps to implement the PCB designs:

1. Designed the circuit
2. Confirmed availability of all values of the components.
3. Verification:
    - Level I: Checked the design using simulation program
    - Level II: Checked the design on the breadboard
4. Drew the circuit schematic using **OrCAD** Capture
5. Pin connections verification in schematic
6. Created the layout of the circuit using the appropriate footprint for every component
7. The option of creating our own library and constructing its footprint is provided in case of not finding the appropriate footprints for the components among the standard libraries.
8. Added created libraries to the built-in ones.
9. Constructed a footprint for each component by using its dimensions described in the datasheet or measuring it manually, knowing that surface mounted ICs need Special type of pads in footprints
10. Routed the circuit with single-layer-tracks using *automatic routing*
11. Fixed some tracks using *manual routing*

5.8.1 Schematic

5.8.1.1Main Circuit

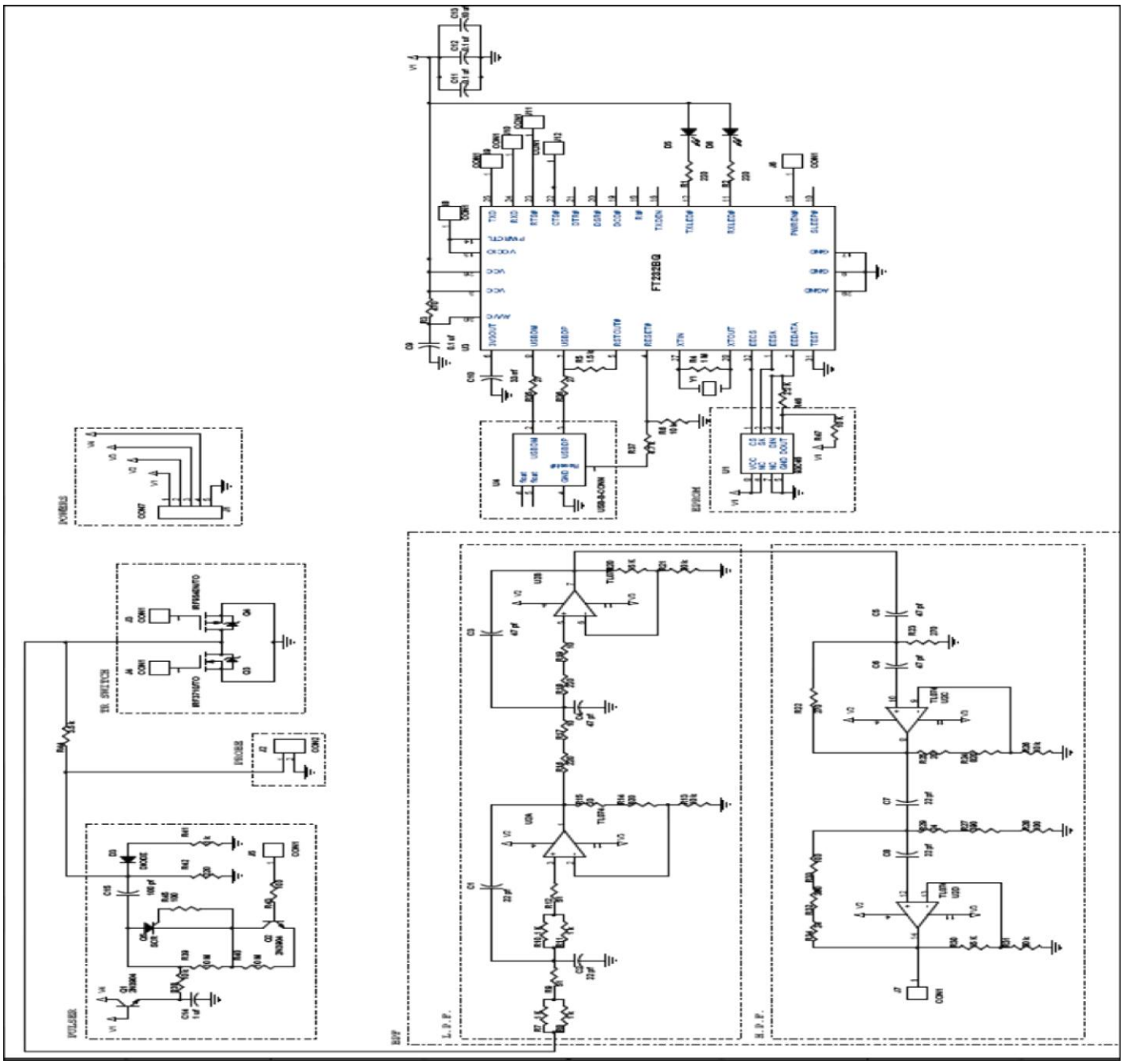


Figure 5.8-1 Overall Schematic of System

5.8.1.2 Pulser, Probe & TR Switch

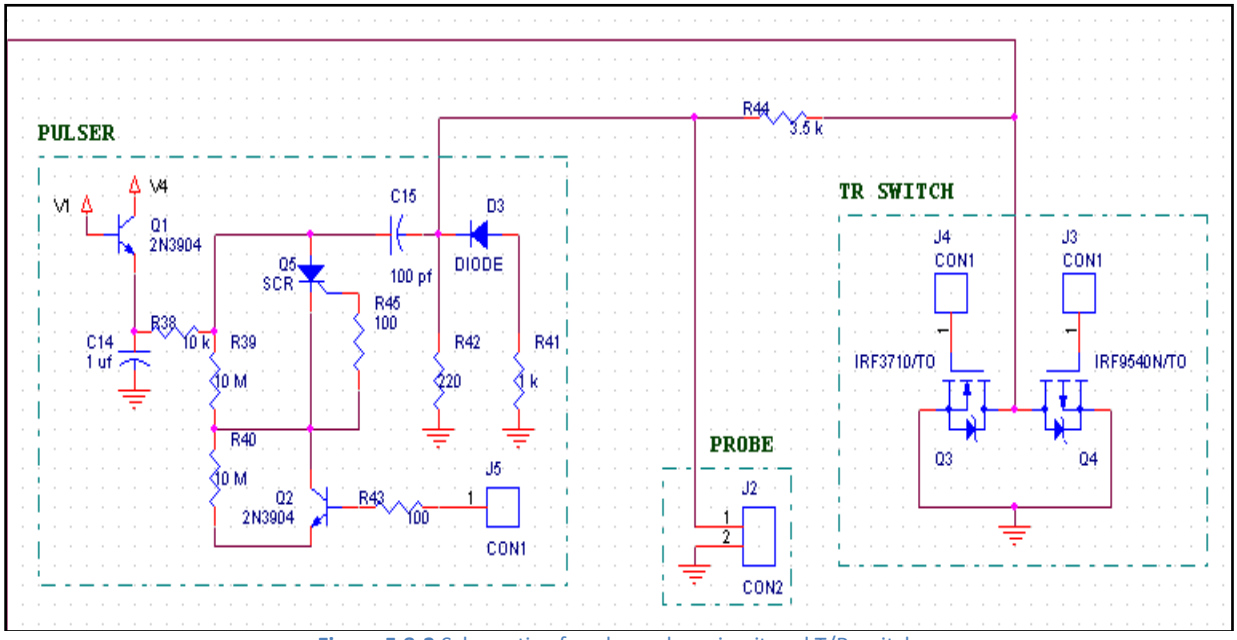


Figure 5.8-2 Schematic of probe, pulser circuit and T/R switch

5.8.1.3 FT232BQ, EPROM & USB connector

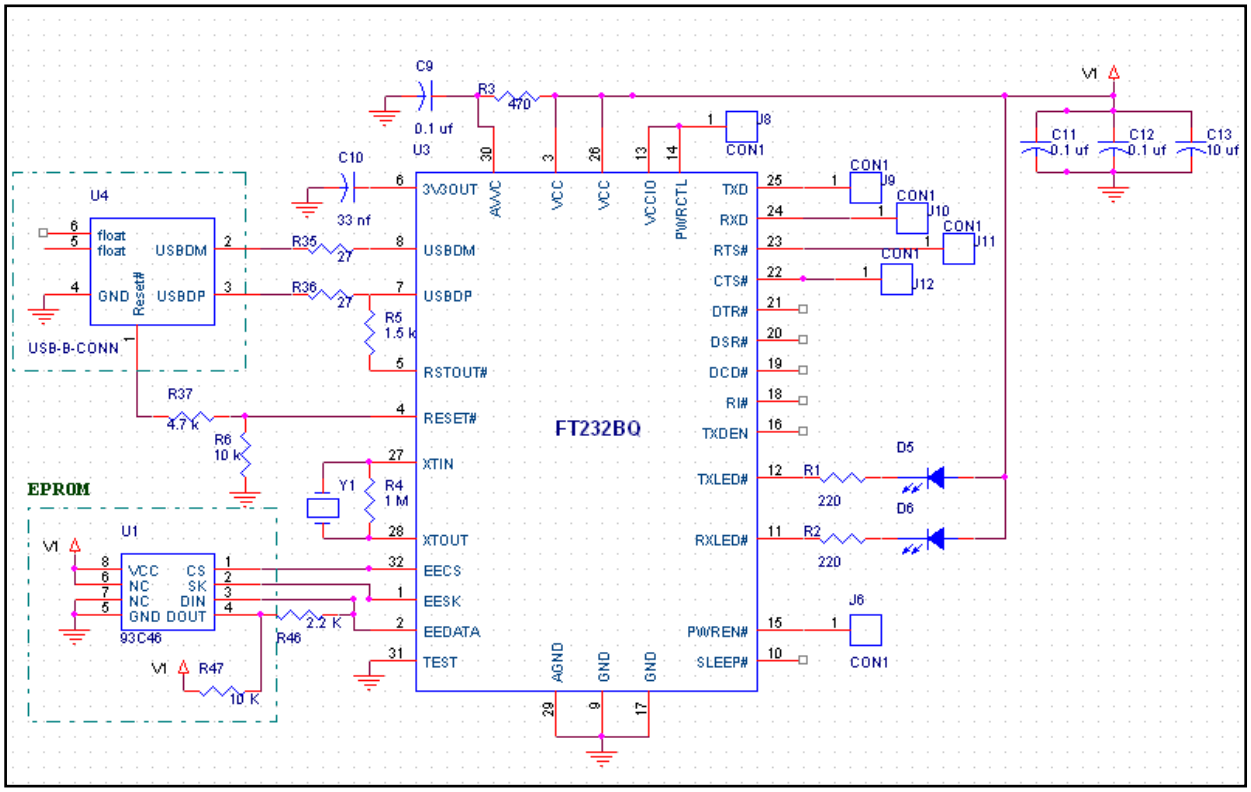


Figure 5.8-3 FT232BQ schematic

5.8.1.4 Band Pass Filter

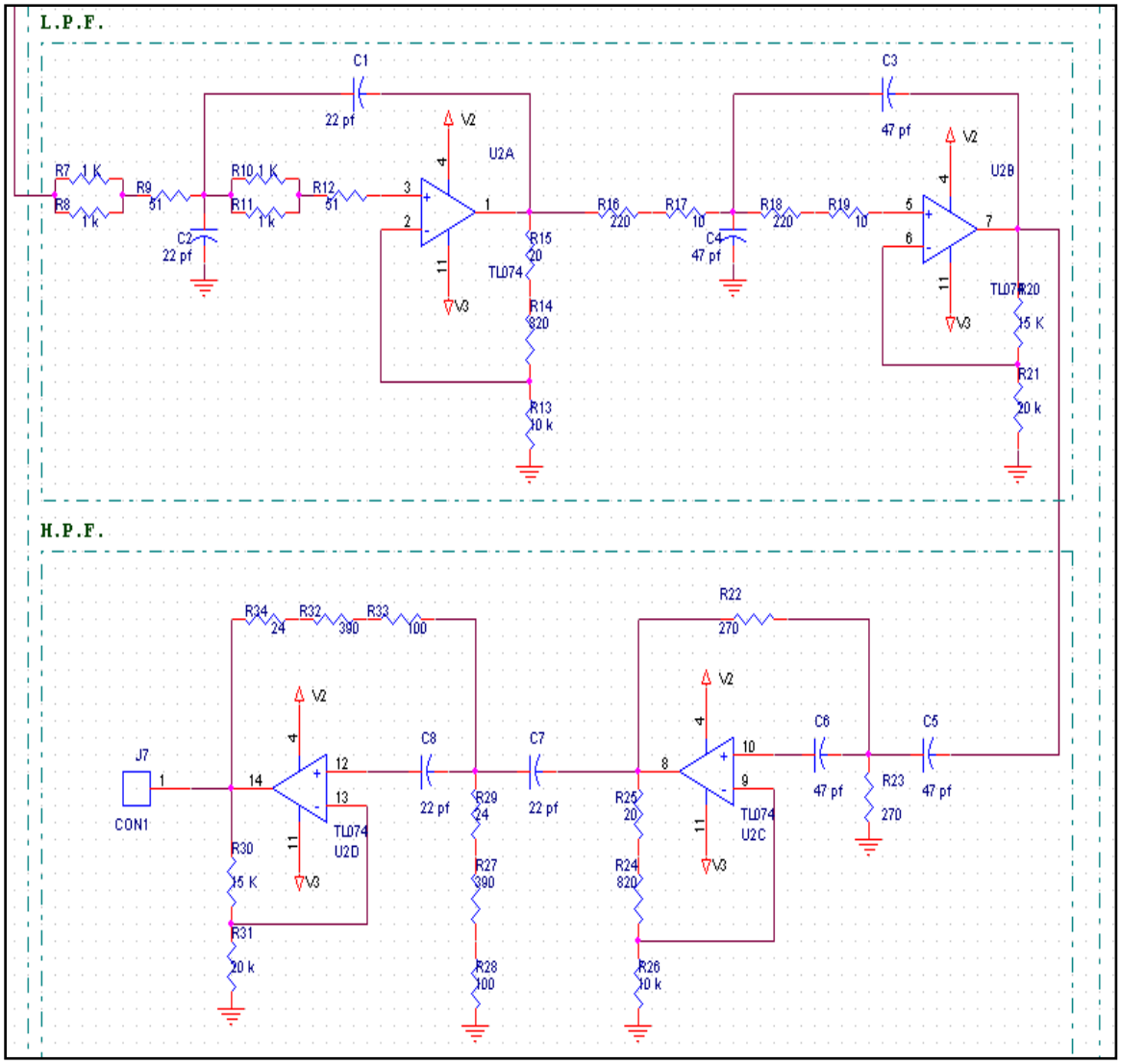


Figure 5.8-4 Bandpass filter schematic

5.8.1.5 Power Supply

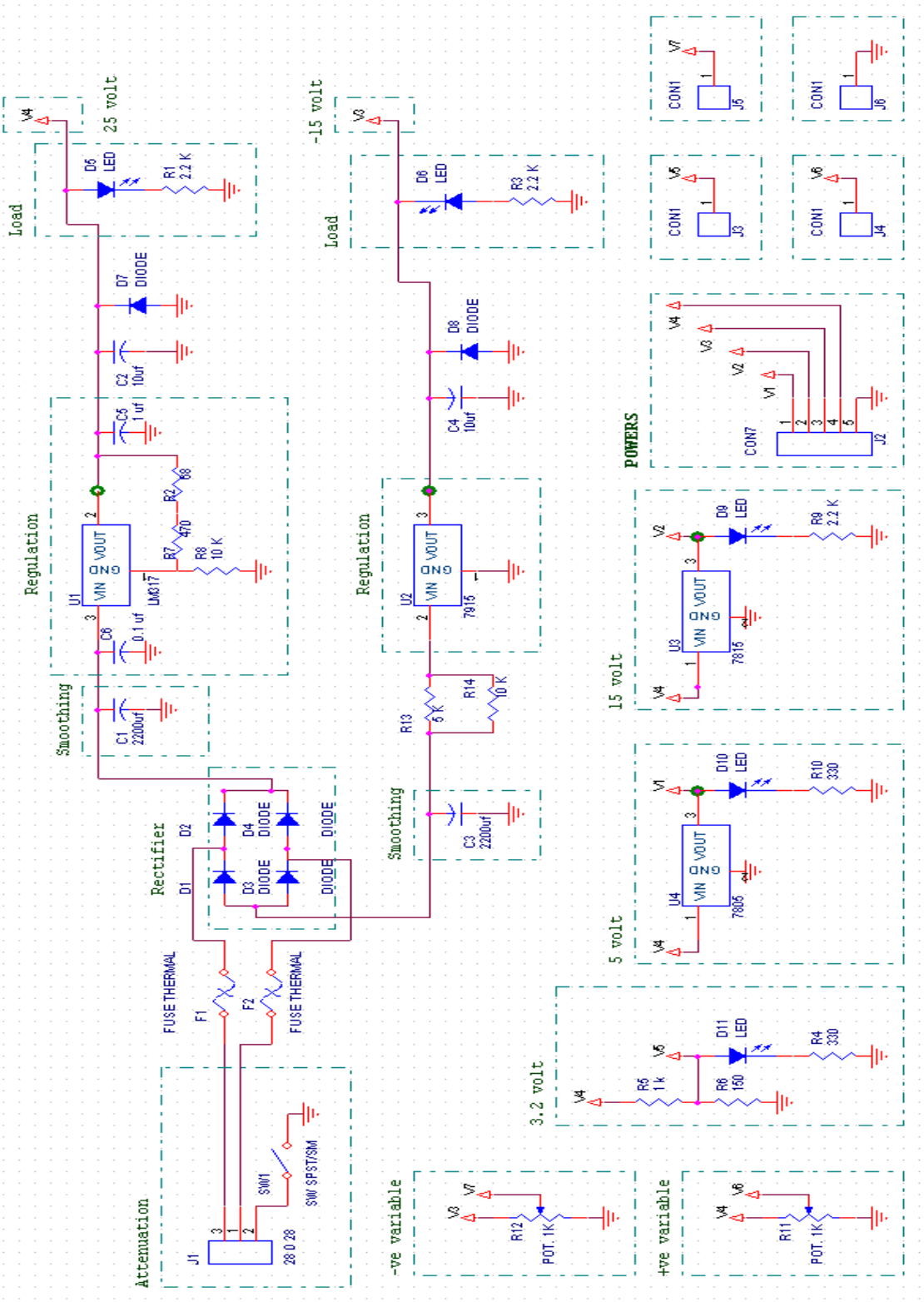


Figure 5.8-5 Power Supply Schematic

5.8.2 Layout

5.8.2.1 Main Circuit

- **Top**

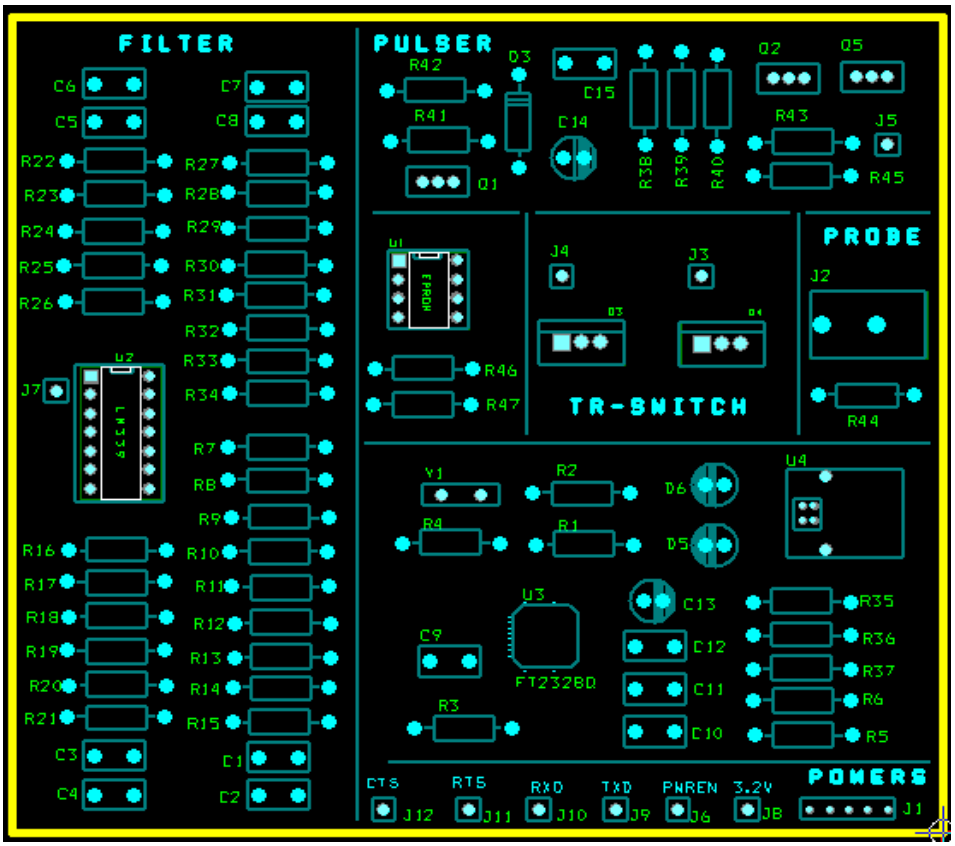


Figure 5.8-6 System PCB Layout, Top view

- **Bottom:**

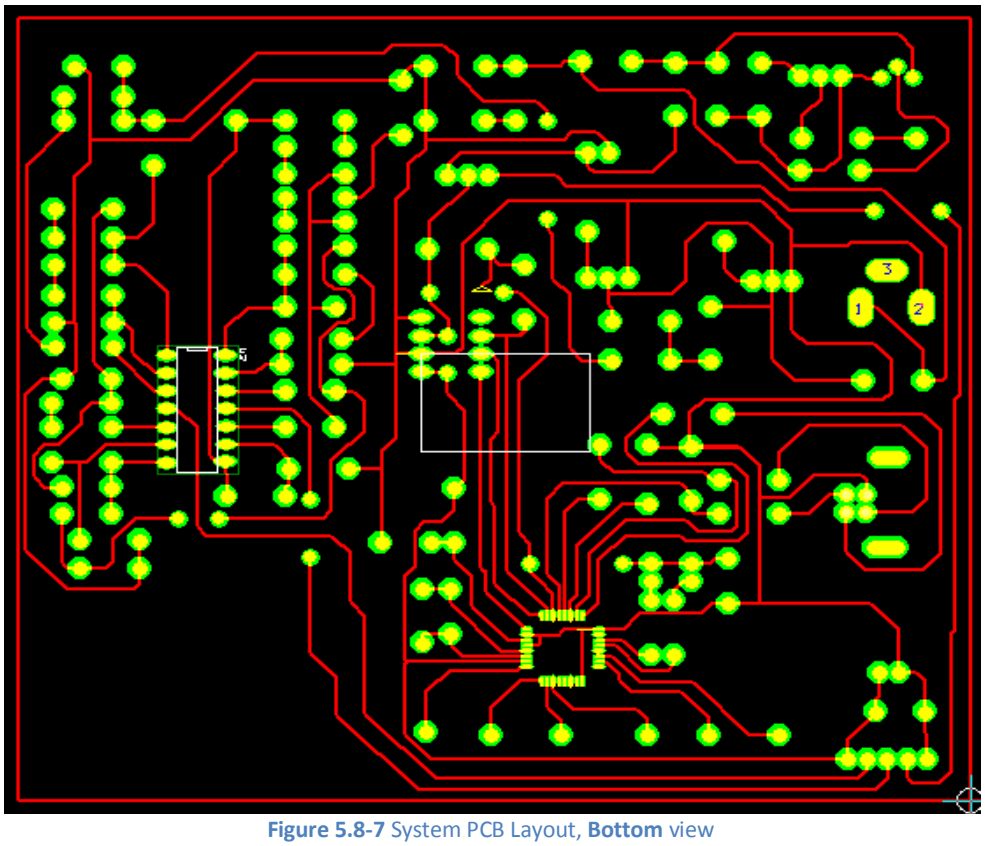


Figure 5.8-7 System PCB Layout, **Bottom** view

5.8.2.2 Power Supply

- **Top:**

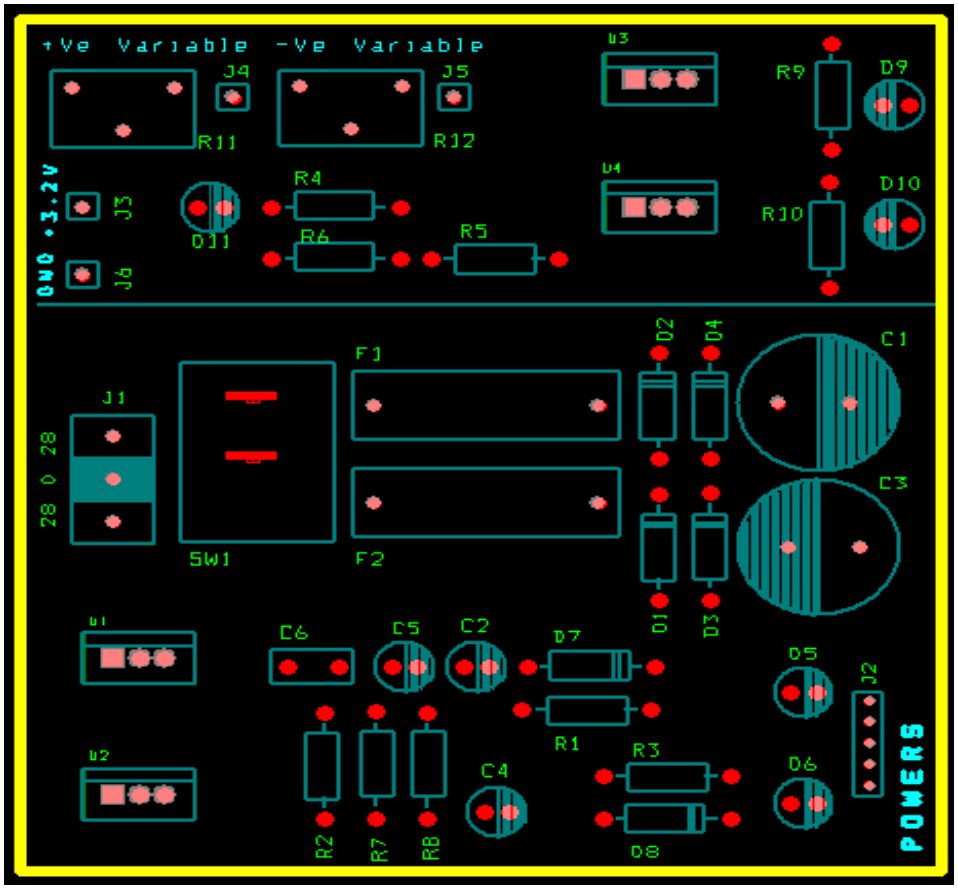


Figure 5.8-8 Powers Supply PCB **Layout**, Top view

- **Bottom:**

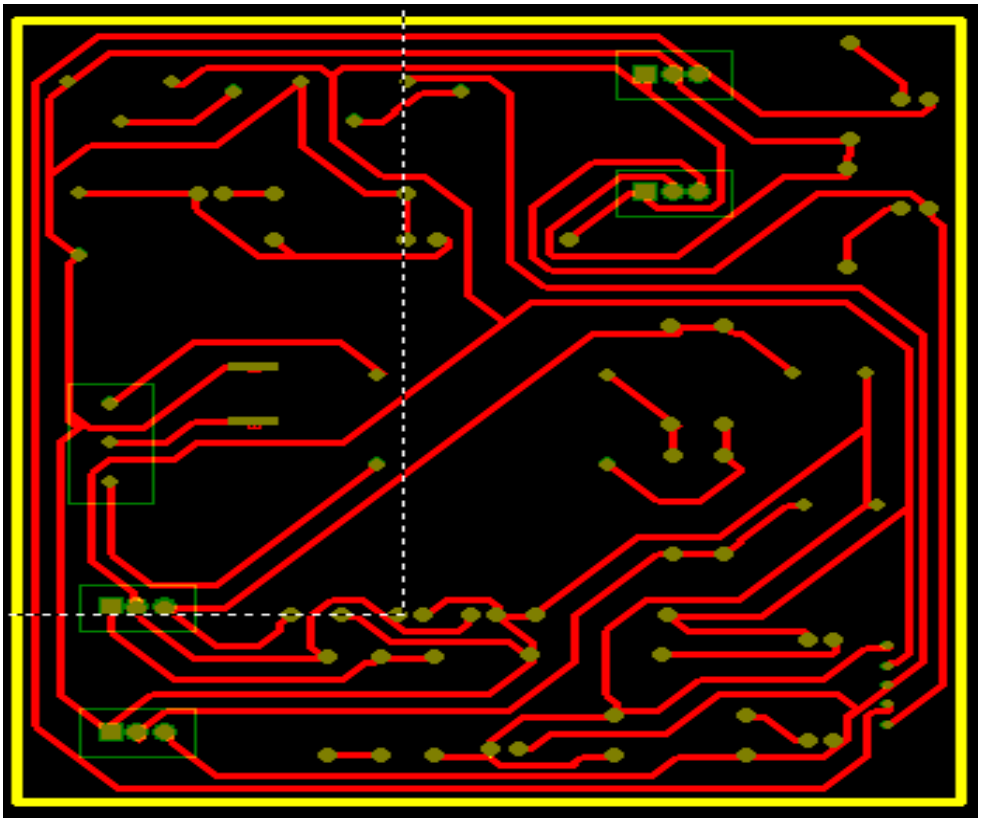


Figure 5.8-9 Power Supply PCB Layout, **Bottom** view

5.9 References

[1] Richard G. Lyons , Prentice Hall Understanding Digital Signal Processing , 2001
[2] Kraig Mitzner, Complete PCB Design Using OrCAD Capture and Layout

Chapter

6

Software

Introduction
Description
Implementation
Program Structure
Methods and Variables
Refernces

6.1 Introduction

Parallel with the Hardware branch discussed in chapter 5 proceeds the software branch. This chapter shows how the Java Platform supports the implemented hardware to achieve the set of requirements of e-DIOLA scan.

6.2 Description

Designed by Java netbeans 6.5. the E-DIOLA Scan software has a user friendly, interactive graphical user interface making interaction extremely easy and straight forward. The continuously appearing instructions in the window guides the user through the program. This facilitates the usage and saves the time taken to explore the software.

To make a measurement, the operator first displays the acquisition screen, and follows the instructions for its various controls. The probe is applied to the patient's eye directly.

For biometry, there are two modes of operation, automatic and manual. In automatic mode the system identifies the critical structures in the eye and makes the measurements. In manual operation the operator must freeze an image and then select the points for the measurements. There are optional settings for particular cases such as dense cataracts or silicone filled eyes.

The user is first welcomed with an attractive welcome screen. Upon clicking onto the screen the user enters the software.

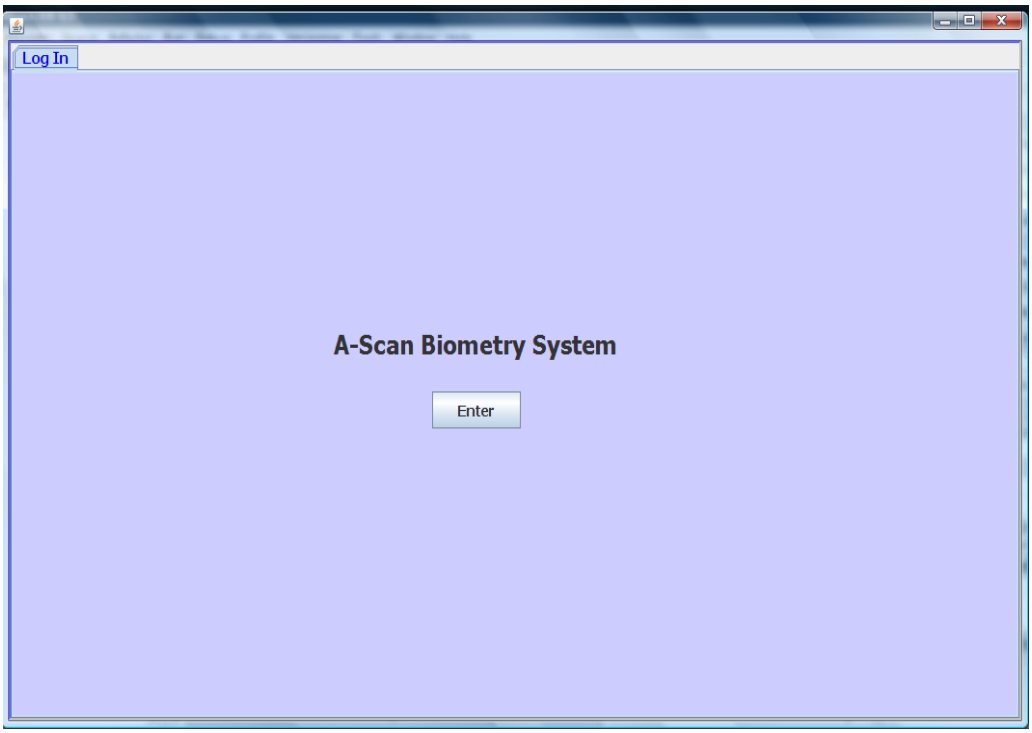


Figure 6.2-1 Login Tab

Usually the ophthalmologist would primarily add a record for the patient by filling a form with the patient's personal information.

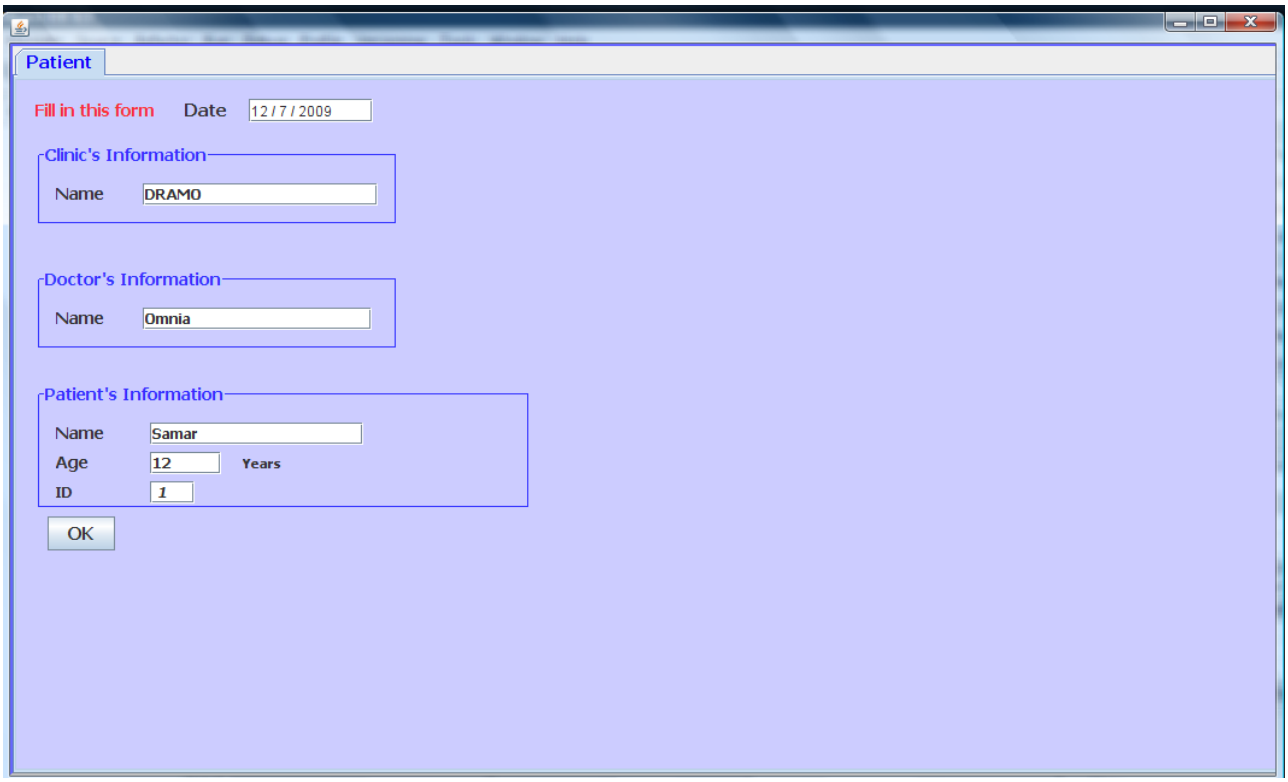


Figure 6.2-2 Patient Information Tab

To run a scan, the user only needs to click on the A-Scan tab and check the radio button that correspond to the patient's case.

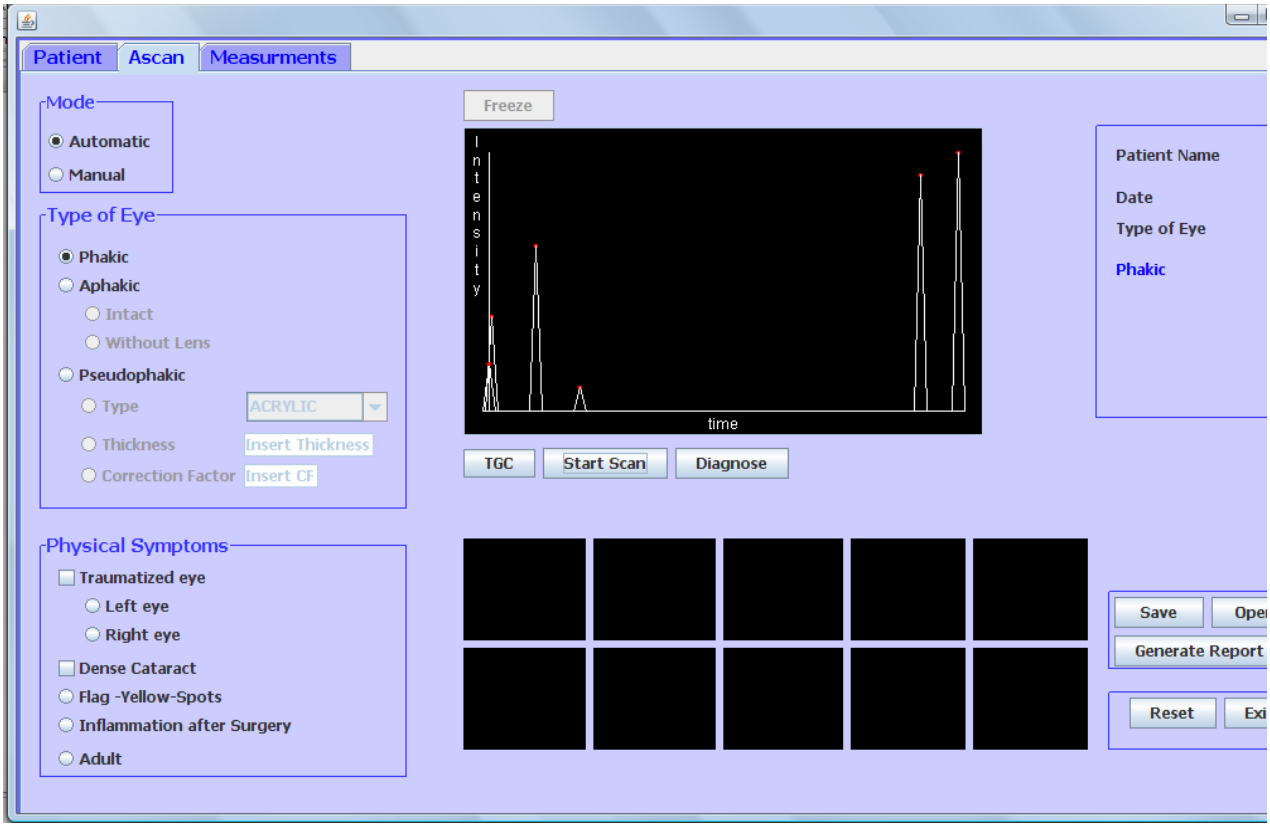


Figure 6.2-3 A-Scan Tab

The Auto-diagnosis tab shows the opthalmologist what disease e-DIOLA has detected automatically.

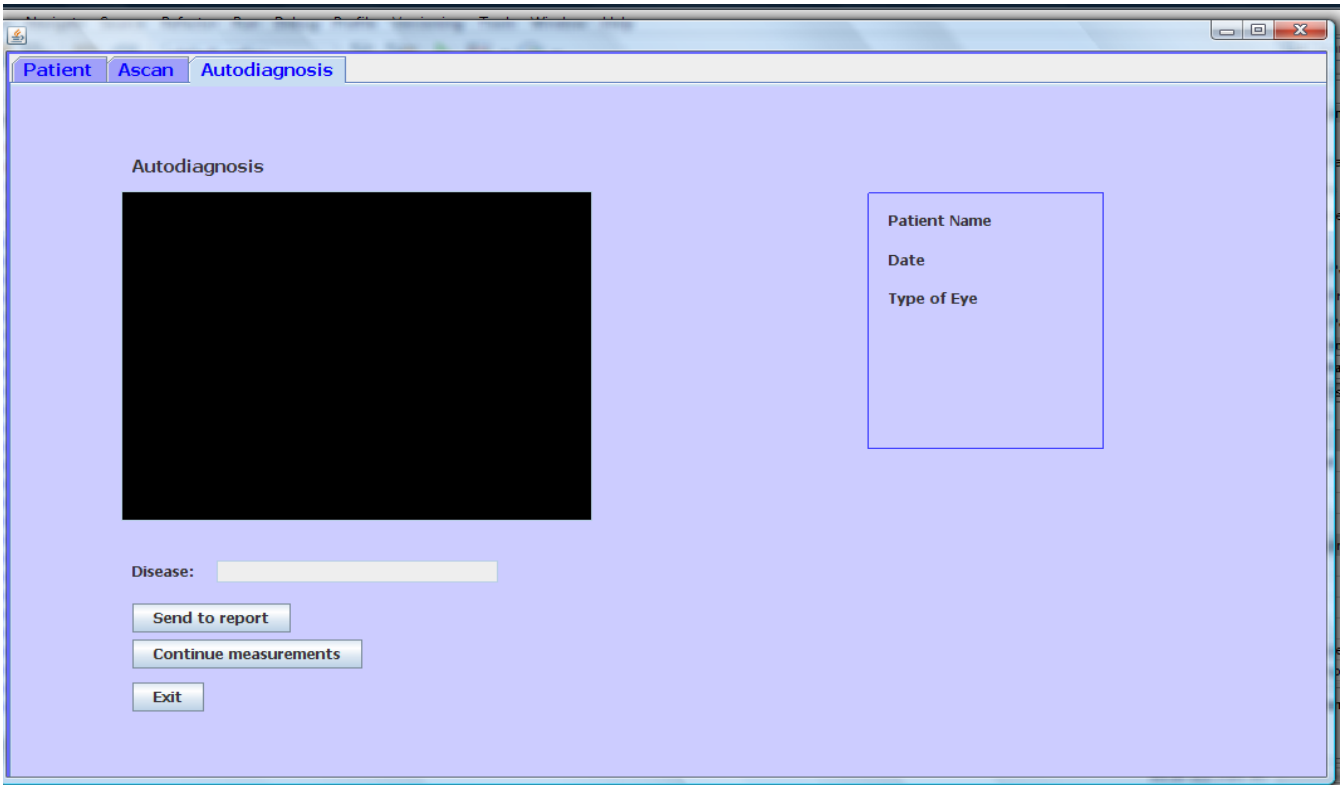


Figure 6.2-4 Autodiagnosis Tab

The results of the test are published in form of a printable report, thus preventing loss of data.

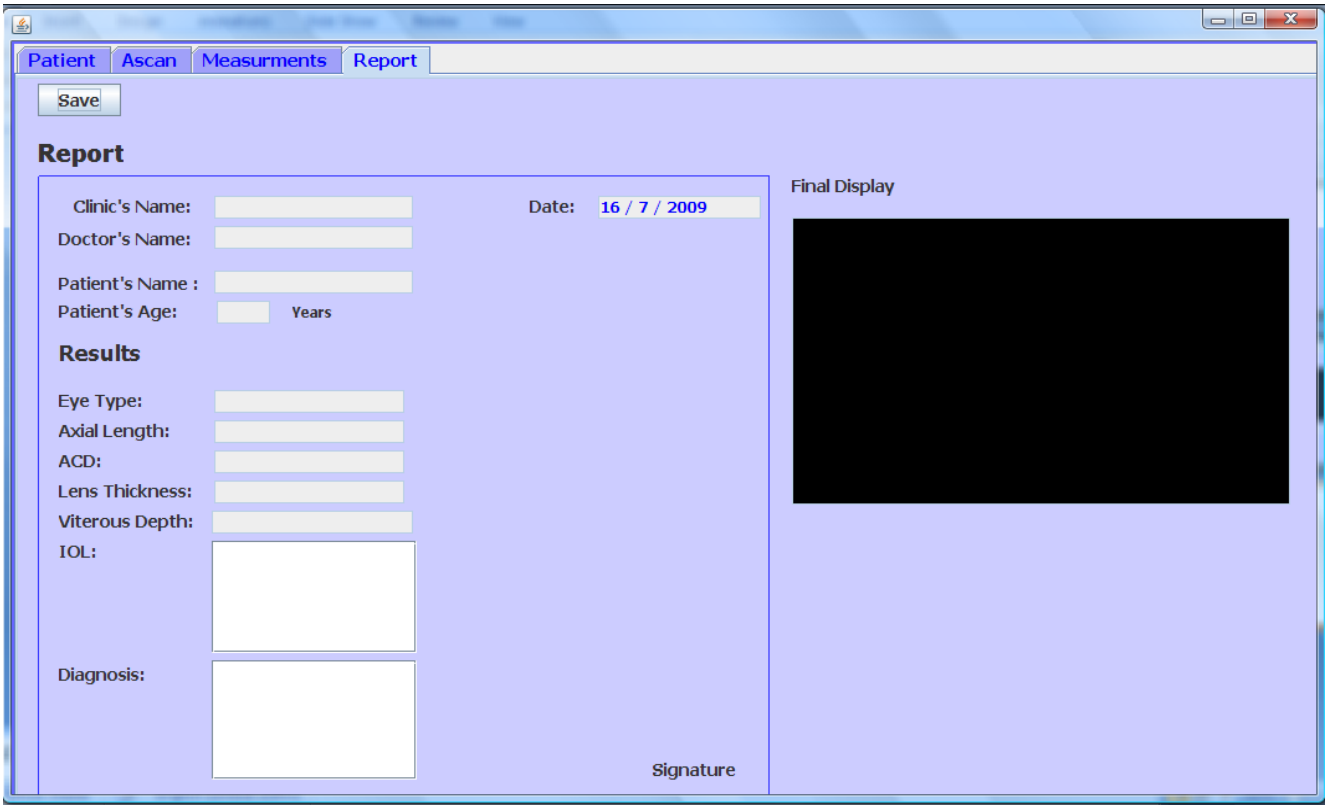
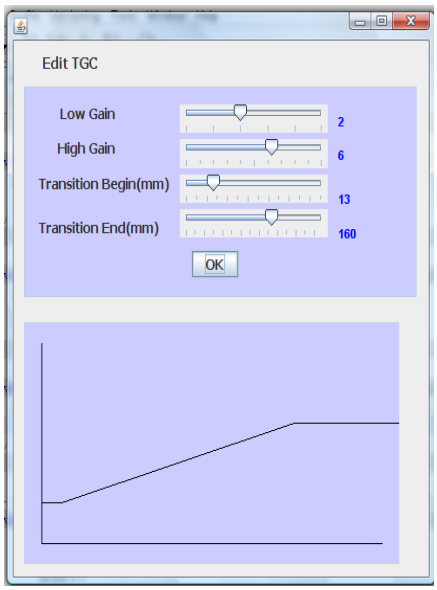


Figure 6.2-5 Patient Report Generation Tab

The ophthalmologist can control the TGC curve if he/she is operating in the manual scan mode.
The first slide presents the value of the initial (low) gain and the second slider sets the values for the High gain. The other two determine the slope by which attenuation will be compensated for.



6.2-6 TGC Cotrols and theresulting curve

IOL Formulas are specific to the axial length. It is one of e-DIOLA's features, that the software automatically decides the most suitable formula for the case at hand. At the same time the user is given the option to choose the formula manually.

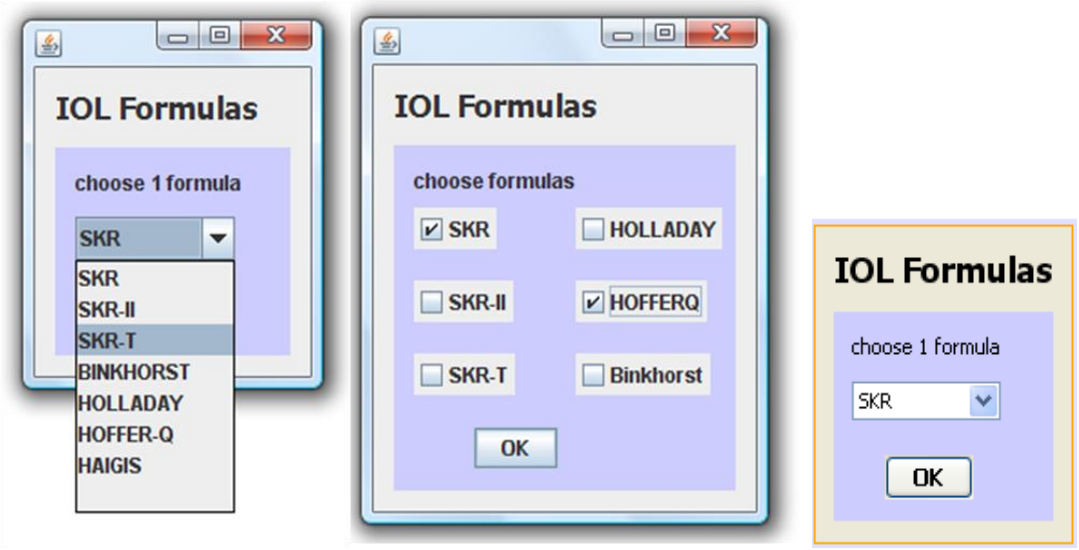
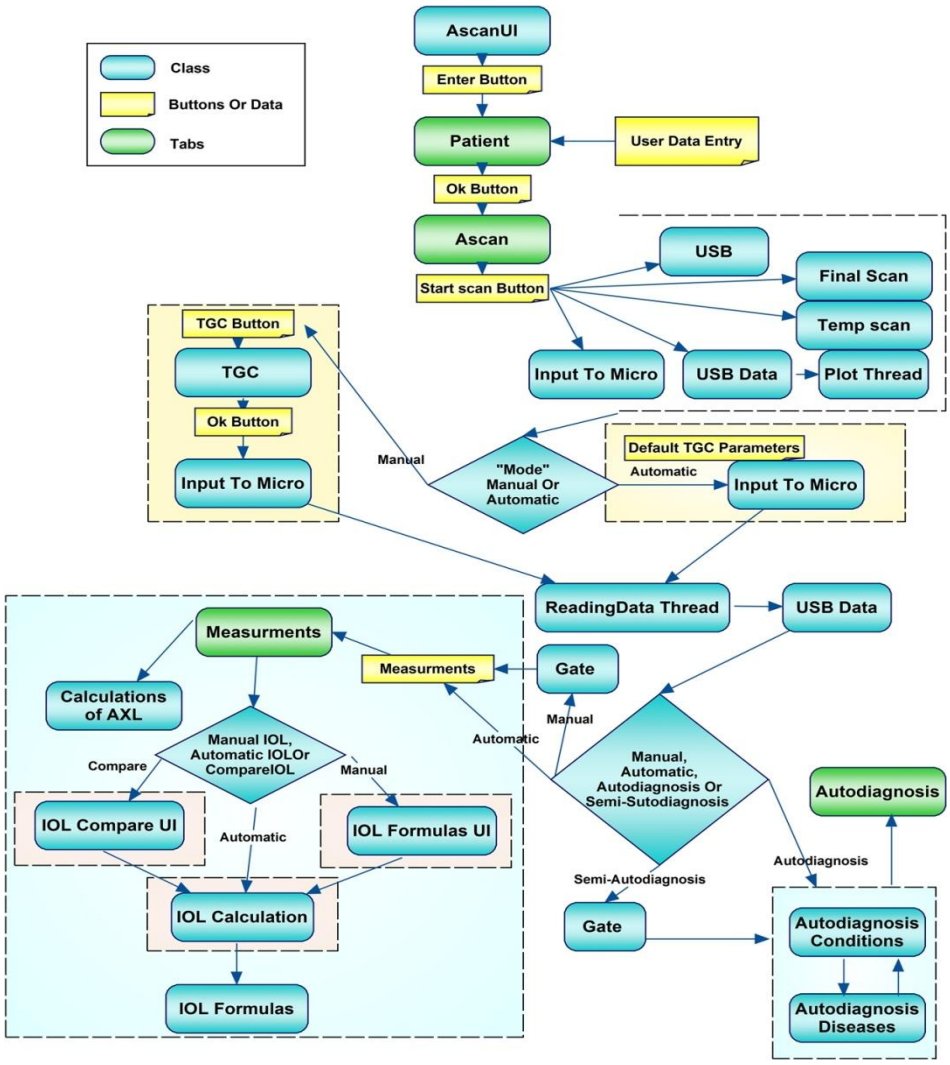


Figure 6.2-7 IOL Formula Selection

6.3 Implementation

6.3.1 Program Structure



6.3-1 Class Diagram of Java Code

Class AscanUI
The main class that contains all the principal taps.

Class Ascan_Help
Contains instructions that help the user to deal easily with DIOLA software.

Class AutoDiagnosis_Conditions
Contains undefined peaks and returns the accurate diseases to GUI.

Class AutoDiagnosisDiseases
Contains all cases to detect different diseases.

Class Avg_StdDev
Contains functions that calculate the average and standard deviation of array of values.
Class Calculations
Contains functions that calculate the axial length, ACD, lens thickness and viterous depth.

Class Edit_TGC
GUI that enables the user to change the values of a,b,c and d of the TGC.
Where:

c=lower gain value
d=upper gain value
a=end of zero initial interval
b=end of sloped gain interval

Class Final_Scan
Contains arrays that contain all data about the final scan (e.g. peaks numbers, intensities and times of peaks).

Class Gate
This class is created in the manual mode to enable the user choose the peaks and name them manually.

Class Inputs_To_Micro
Contains functions that convert data that will be sent to microcontroller to byte equivalent to the agreed code.

Class IOL_Calculation
Contains functions that apply the algorithm of calculating IOL automatically, manually or compare between two or more IOL formulas results.

Class IOL_Compare_UI
GUI that enables the user to choose the formulas of IOL he wants to compare their results.

Class IOL_Formulas_Calc
Contains function for each formula that calculate the corresponding IOL.

Class IOL_Formulas_UI
GUI that enables the user to choose manually only one formula of IOL.

Class No_Of_Peaks
Contains functions that return some information about the peaks like number of peaks and names of peaks.

Class Plot
This class for plotting the scan of peaks in the graph window by applying some scaling in x and y direction.

Class Reading_Thread
Reading data come from microcontroller.

Class Save
Saving patient report as file document or image and display result as an image.

Class Temp_Scan
Contains arrays that contains all temporary data about the scan (e.g. peaks numbers, intensities and times of peaks).

Class USB
Searching for the available USB device and getting its input and output streams.

Class USB_Data
In this class we apply the protocol by which we can write and read from the microcontroller.

6.3.2 Methods and Variables

6.3.2.1. Class AscanUI

main
public static void main(java.lang.String[] args)
Parameters:
args - the command line arguments
Creates new form from AscanUI.

6.3.2.2. Class Ascan_Help

main
public static void main(java.lang.String[] args)
Parameters:
args - the command line arguments
Creates new form from Ascan_Help.

6.3.2.3. Class AutoDiagnosis_Conditions

check_arrays

public void check_arrays()
throws java.lang.Exception

Throws:

java.lang.Exception

Returns names of different diseases in multiple regions

6.3.2.4. Class AutoDiagnosisDiseases

1. MAXAMP

public float MAXAMP(float[] Amplitude)

Returns maximum element of the array.

2. Start_End

public java.lang.String[] Start_End(int i)

Returns the surround peaks of certain region.

3. Construct_Arrays

public float[][] Construct_Arrays(int i,
float[] intensity,
float[] Time)

Returns array of undefined peaks intensity and time.

4. Vitreous_Diseases

public java.lang.String Vitreous_Diseases(float M_Amp,
float[] Vamp,
float[] Vtime)

Returns possible disease due to certain conditions in vitrous region.

5. Sclera_Disease

public java.lang.String Sclera_Disease(float M_Amp,
float[] Samp)

Returns possible disease due to certain conditions in sub- sclera region.

6. Retinal_Disease

public java.lang.String Retinal_Disease(float[] Ramp,
float[] Rtime,
int G,
float ACamp,
float PCamp,
float PLamp,
float ALamp,
float Ratamp,
float Samp)

Returns possible disease due to certain conditions in sub-retinal region.

6.3.2.5. Class Avg_StdDev

1. Avg

public float[] Avg(float[][] time,
int n,
int No_Of_peak)
throws java.lang.Exception

Throws:
java.lang.Exception
Returns one dimensional array of the average value of each column in two dimensional array of values.

2. Std

public float[] Std(float[][] time,
int n,
int No_Of_peak)
throws java.lang.Exception

Throws:
java.lang.Exception
Returns one dimensional array of the standard deviation value of each column in two dimensional array of values.

6.3.2.6. Class Calculations

1. Axial_Lens

public float Axial_Lens(java.lang.String Technique,
int PNo,
float[] Time,
java.lang.String Eye_Type)

Returns axial length of the eye according to the type of eye
And special calculations done on pseudophakic “Lens type,Thickness or Correction factor”

2. Type_Technique

public float Type_Technique()
Returns the thickness of the lens due to the type of it.

3. Thickness_Technique

public float Thickness_Technique()

Returns the thickness of the lens that inserted.

4. ACD

public float ACD(java.lang.String Lens_Type,
float[] Time,
java.lang.String Technique,
int PNo)

Returns the Anterior Chamber Depth only for phakic and pesudophakic eye

5. LT
public float LT(java.lang.String Lens_Type,
java.lang.String Technique)
Returns the length thickness only for phakic and pesudophakic eye

6. VD
public float VD(java.lang.String Lens_Type,
java.lang.String Technique,
int PNo)
Returns the vitrous depth only for all types of eye

6.3.2.7. Class Edit_TGC
1. draw
public void draw()
Draws the TGC graph with the values a,b,c and d got from the sliders.

2. main
public static void main(java.lang.String[] args)
Parameters:
args - the command line arguments
Creates new form from Edit_TGC.

6.3.2.8. Class Final_Scan

6.3.2.9. Class Gate
1. Gate
public void Gate(java.awt.Point P,
int Peak_No,
int Sacn_No)
Consists of the two functions: “Get_Position_Of_Peak” and “Insert_Peak_Into_array_Manual” which will be discussed next.

2.Get_Position_Of_Peak
public float[] Get_Position_Of_Peak(java.awt.Point P)
Selects one peak from the array which is the nearest one to that the user clicked on the screen by searching in the array of peaks with some ranges.

3. Insert_Peak_Into_array_Manual
public void Insert_Peak_Into_array_Manual(float[] XY,
int Peak_NO,
int Sacn_No)

Inserts the selected peak into the two dimensional array of rows indicate number of scans and columns indicate number of peaks.

4. Saving

public void Saving()

Saves the names of peaks in array with putting name “x” to the unselected peaks which will be used in the autodiagnosis.

5. Set_ToolTip

public java.lang.String Set_ToolTip(java.lang.String Symbol)

Sets a tooltip with the name of peak that the user should select it next.

6.3.2.10. Class Inputs_To_Micro

1. Inputes

public byte Inputes(boolean D_C,
int A_M,
java.lang.String Lens_Type,
java.lang.String Material)

Converts inputs data that will be sent to microcontroller to byte by returning one byte equivalent to the combination of some inputs.

2. FRC

public byte FRC(boolean F,
boolean R,
boolean S)

Converts Freeze, Reset and Continue signals that will be sent to microcontroller to byte by returning one byte equivalent to the combination of them.

3. TGC

public byte[] TGC(float a,
float b,
float c,
float d)

Takes the float values of a,b,c and d of the TGC and casting the them to array of bytes to be read when send to microcontroller.

6.3.2.11. Class IOL_Calculation

1. AutoIOL

public float[] AutoIOL(float AL,
float K1,
float K2,
float Ref,
AscanUI A)

Applies the algorithm of calculating IOL automatically by choosing the best formula of IOL due to type of eye and axial length.

2. Manual_IOL

public float Manual_IOL(float AL,
float K1,
float K2,
float Ref,
java.lang.String formula)

Applies the algorithm of calculating IOL manually from the chosen formula by the user.

3. Compare

public float[] Compare(float AL,
float K1,
float K2,
float Ref,
java.lang.String[] formulas)

Applys the algorithm of Comparing between two or more IOL formulas results.

6.3.2.12. Class IOL_Compare_UI

main

public static void main(java.lang.String[] args)

Parameters:

args - the command line arguments

Creates new form from IOL_Compare_UI.

6.3.2.13. Class IOL_Formulas_Calc

1. SKR

public float SKR(float AL,
float K1,
float K2)

Calculates the IOL using SKR formula.

2. Binkhorst

public float Binkhorst(float AL,
float K1,
float K2,
float Ref)

Calculates the IOL using BINKHORST formula.

3. SKR2

public float SKR2(float AL,
float K1,
float K2)

Calculates the IOL using SKR2 formula.

4. HofferQ

public float HofferQ(float AL,
float K1,
float K2,
float Ref)

Calculates the IOL using HofferQ formula.

5. Holladay

public float Holladay(float AL,
float K1,
float K2,
float Ref)

Calculates the IOL using Holladay formula.

6. SKRT

public float SKRT(float AL,
float K1,
float K2,
float Ref)

Calculates the IOL using SKRT formula.

6.3.2.14. Class IOL_Formulas_UI

1. main

public static void main(java.lang.String[] args)

Parameters:

args - the command line arguments

Creates new form from IOL_Formulas_UI.

6.3.2.15. Class No_Of_Peaks

1. Get_No_Of_Peaks

public int Get_No_Of_Peaks(java.lang.String Eye_Type)

Returns the actual number of peaks corresponding to the eye type that the user input it initially.

2. Naming_Peaks

public java.lang.String Naming_Peaks(int Peak_NO,
java.lang.String Eye_Type)

Returns the name of peak when chosen by the gate in case of using manual mode.

6.3.2.16. Class Plot

1. run

public void run()

Specified by:

run in interface java.lang.Runnable

Execute the plotting function.

2. plotting

public void plotting(float[] time,
float[] intensity)

plotting the scan of peaks in the graph window by applying some scaling in x and y direction, draw coordinates and labels of coordinates.

6.3.2.17. Class Reading_Thread

run

public void run()

Specified by:

run in interface java.lang.Runnable

Reads data come from microcontroller all the time and executes some function due to the data received.

6.3.2.18. Class Save

convert

public void convert(javax.swing.JPanel P,
java.lang.String T)

Convert all panel components and texts to image as jpg

2.19. Class Temp_Scan

2.20. Class USB

GettingStreams

public void GettingStreams()

Gets the input and output streams of the available USB device to be used in writing and reading from the microcontroller.

6.3.2.21. Class USB_Data

1. Read_Data

public void Read_Data(byte[] b)

Reads the array of bytes from the microcontroller with a certain protocol.

2. Write_Data

public byte[] Write_Data(int status,
byte inputs,
byte[] TGC,
byte F_R_C)

Constructs the array of bytes that will be written to the microcontroller with a certain protocol.

3. ToInt

public int ToInt(byte[] b)

Converts the two bytes to the equivalent integer number.

4. ToTime

public float ToTime(int b)

Converts the index of the time which is an integer number come from the microcontroller to the equivalent time with seconds using some calculations

6.4 Validation

6.4.1 Serial Communication

Just upon connecting the USB connector to the Laptop, the OS recognizes the new device and installs its driver. When this is done successfully then the Java Platform easily gives back the device's decription and default ID number. This is shown in Figur 6.4-1.

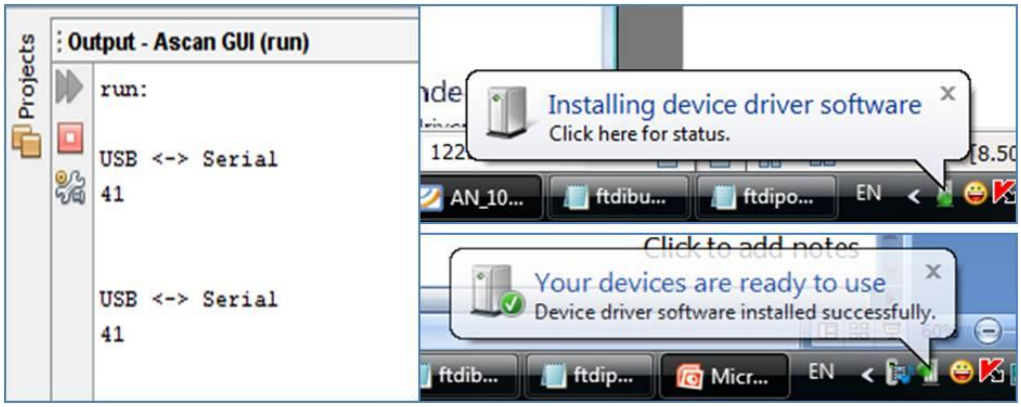


Figure 6.4-1 Successful installation procedure

6.4.2 Plotting Data

As soon as the Java Platform receives data bytes from the MCU they are plotted in the scan window, as shown in Figure 6.4 – 2 .

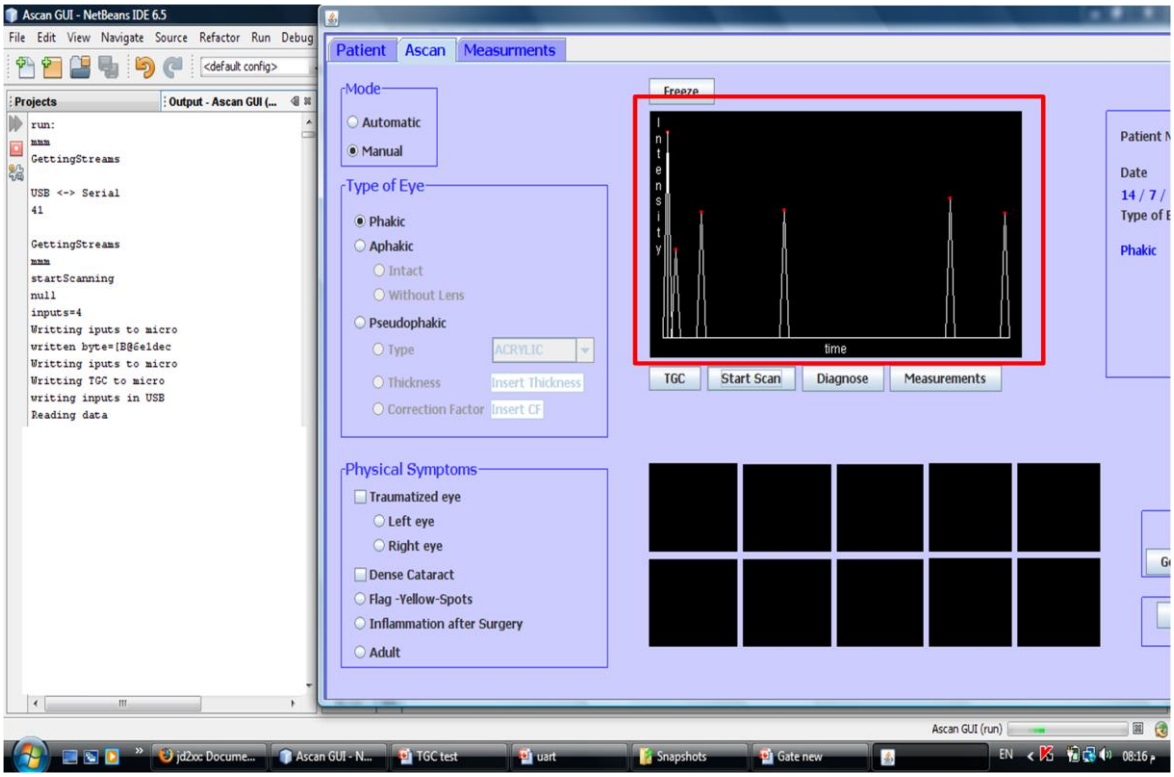


Figure 6.4-2 Plotting the data received from the MCU

6.4.3 IOL and Axial length Measurement

With a single click on the Mesurement button, e-DIOLA automatically performs the necessary calculatons and displays the results in the Results tab.

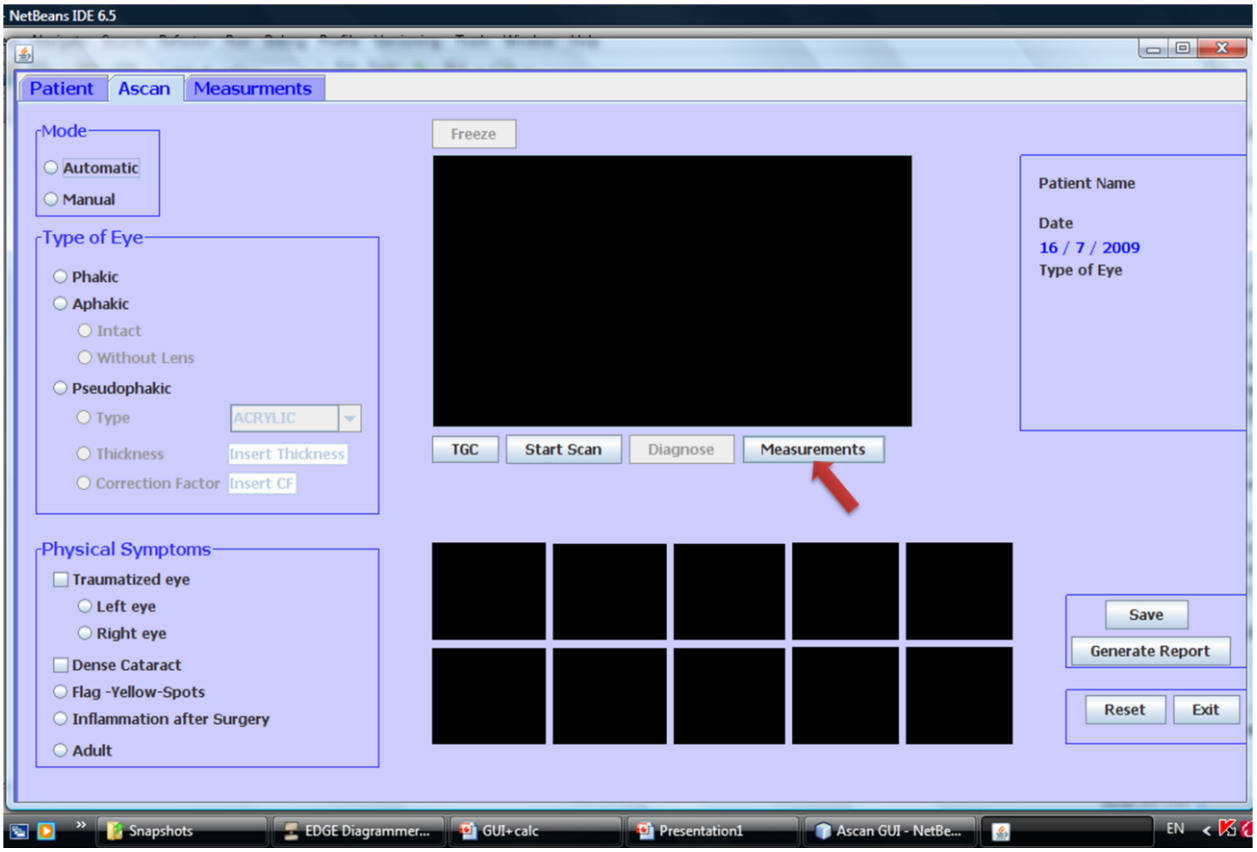


Figure 6.4-3 A-Scan Tab; press Measurement button

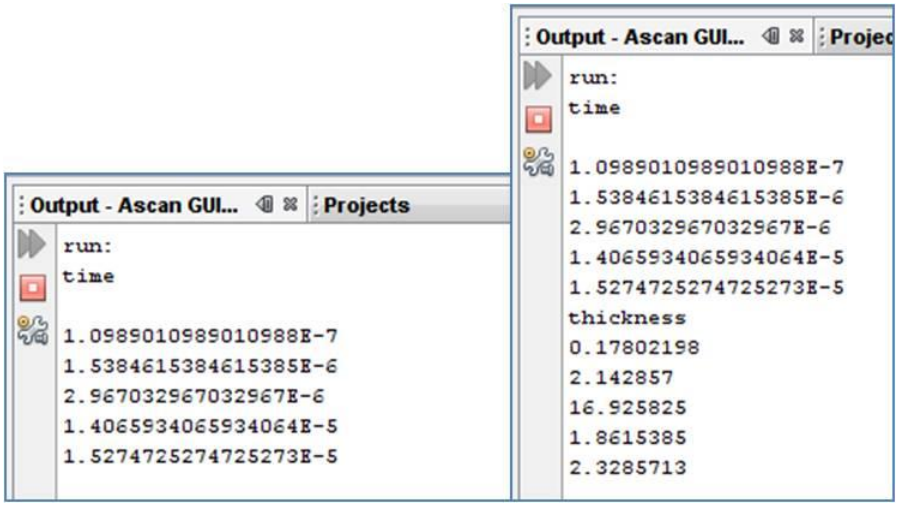


Figure 6.4-4 Calculation of time and thicknesses appear in the command window during debug

The user needs only to type the Keratometry readings in the specified fields. The results will be generated automatically using the best fitting IOL formula.

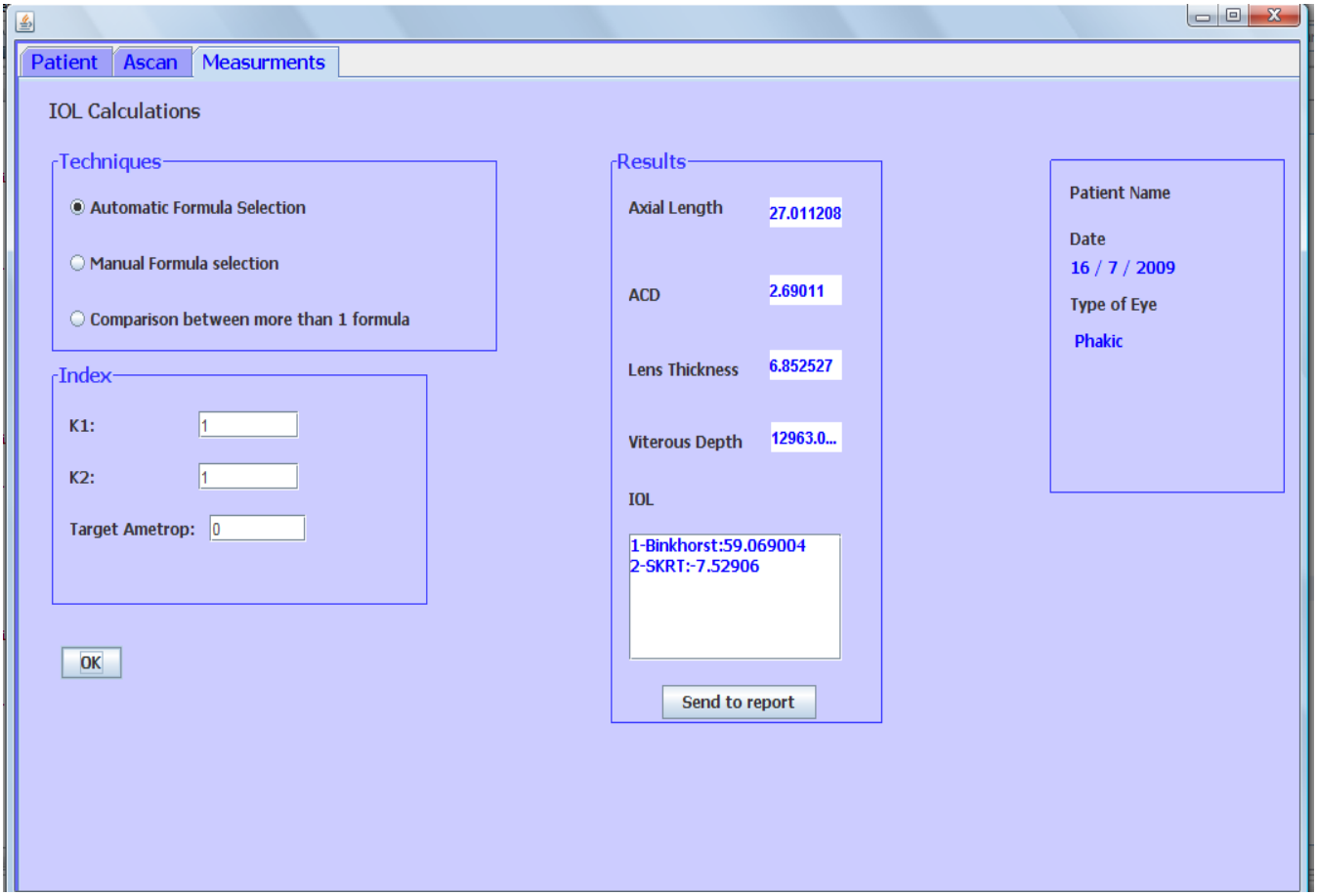


Figure 6.4-5 IOL and Axial length measuremnts are displayed

6.4 References

[1] M.Dietel, J.Dietel, Java-How to program , 4th edition
[2] Java API documentation : http://java.sun.com/j2se/1.5.0/docs/api/
[3] OTI Scan 1000 user manual: User Manual, OTI OPTHALMIC TECHNOLOGIC INC, Software version 5.1, January 2004

Chapter 7

Firmware

Introduction
Memory Model
System Setup
Algorithm Structure
Algorithm Flowchart
Algorithm Functions
Main Data Storage Variables
Main Controlling Flags
Debugging
Performance Analysis
Code Guidelines and Pitfalls
Refernces

7.1 Introduction

The parallel paths of hardware and software shown in Figure 1.6 – 1 meet each other at a certain interface that defines how they communicate with each other. The Firmware of a system provides the necessary set instructions that control this communication. Design and verification of those software and hardware interfaces have become crucial to first-pass success of the overall system design. This chapter will exhaustively demonstrate implantation of e-DIOLA scan Firmware.

7.2 Memory Model

Although **SMALL** memory model generates smallest and fastest possible code, two reasons forced us to use **LARGE** memory model. First, is the multiple dynamically allocated arrays using pointers that we used which already reside in a memory pool section of the external memory and to ensure efficient exchange of data the rest of the runtime variables were to be in the same memory. Second, is that when **SMALL** memory model was used, the runtime Data memory caused overflow.
Internal memory was explicitly used to store all non-pointer variables using **data** directive while the simulation Look-up table used in validation in **Chapter 9** resides in the external memory.

7.3 System Setup

7.3.1 Sampling Frequency

Is adjusted to **4.55 MHz** which according to the fact of our Band limited signal would be sufficient to acquire the signal with fine resolution.

7.3.2 Samples per scan

One of the main challenges we faced was the determination of the acquisition array size. We determined the maximum possible array size depending on two factors
sampling frequency and time of returning echo.
Assuming that the very first spike (anterior cornea) appears at time Zero then the fastest returning echo will from the second spike (posterior cornea) at time 0.24 µseconds (RTT).
The sampling frequency of the ADC is 4.55 MHz, corresponding to a sampling time of 0.219 µseconds which is shorter than the time needed for the fastest echo. In other words, index 1 of the acquiring array will contain the first arriving spike.

$$\text{Index}\,[n](n)(t_{sampling}) = t_{arrival}[n]$$

$$\text{Index}[1] = \frac{\text{tarrival}\,[1]}{\text{tsampling}}$$

$$= \frac{0.24}{0.219}$$

$$= 1.09 \approx 1$$

Thus the first factor is satisfied and the used sampling frequency is suitable.

The latest returning echo comes from the sclera of an aphakic eye (silicon) at time 47.08 µseconds. The largest array size is then:

$$\frac{\text{tmax [µseconds]}}{\text{tsampling [µseconds]}} = \text{Index[max]}$$

$$\frac{47.08}{0.219} = 215$$

Accounting for worst case conditions we set the maximum array size to 260 indices. Thus the second factor is well considered.

7.3.3 System clock

Is adjusted to maximum possible clock frequency **22118400** (external oscillator) to ensure real time acquisition and processing

7.3.4 Baud rate

Is adjusted to 115200 as maximum of UART on kit to ensure fast transmission of scans to Software to be displayed.

7.4 Alogrithm Structure

7.4.1Header Files

Initialization Specific (**Initialization.h**): Includes Function Prototypes of microcontroller initialization including UART, ADC , ports and system clock.
Program Specific **(Ascan.h)** Includes:

1. Global Defined Variables
2. Function Prototypes of all processing functions

Kit Specific **(C8051F020.h)**: Includes SFR and sbit definitions for the MCU kit, some sbits were added to be accessed by name through the code.

7.4.2Modules

Our project is divided to 5 modules, each specific to performing a certain processing functionality.

Main module **(Main.c)** : Includes main running loop and all inirialization functions. It acts as the central ground between all other modules where data can only exchanged between them passing through it. Inretturpt service routines of UART and ADC are included here.

Amplification module **(Amplification.c)**: Includes functions used to amplify signal as a first step after aquizition.

Classification **(Classification.c)**: Includes peak detection and peak amplitude definition functions.

Checking module **(CheckForNormality.c)**: Performs checks on detected spikes including corenal compression, probe misalignment. It gathers 10 valid scans and performs average and standard deviation calculations on them. It also sends final arrays of data to the Software via UART.

Serial modules **(Serial.c)**: Includes array formulation functions and byte-by-byte trasfere through UART.

7.5 Algorithm Flowchart

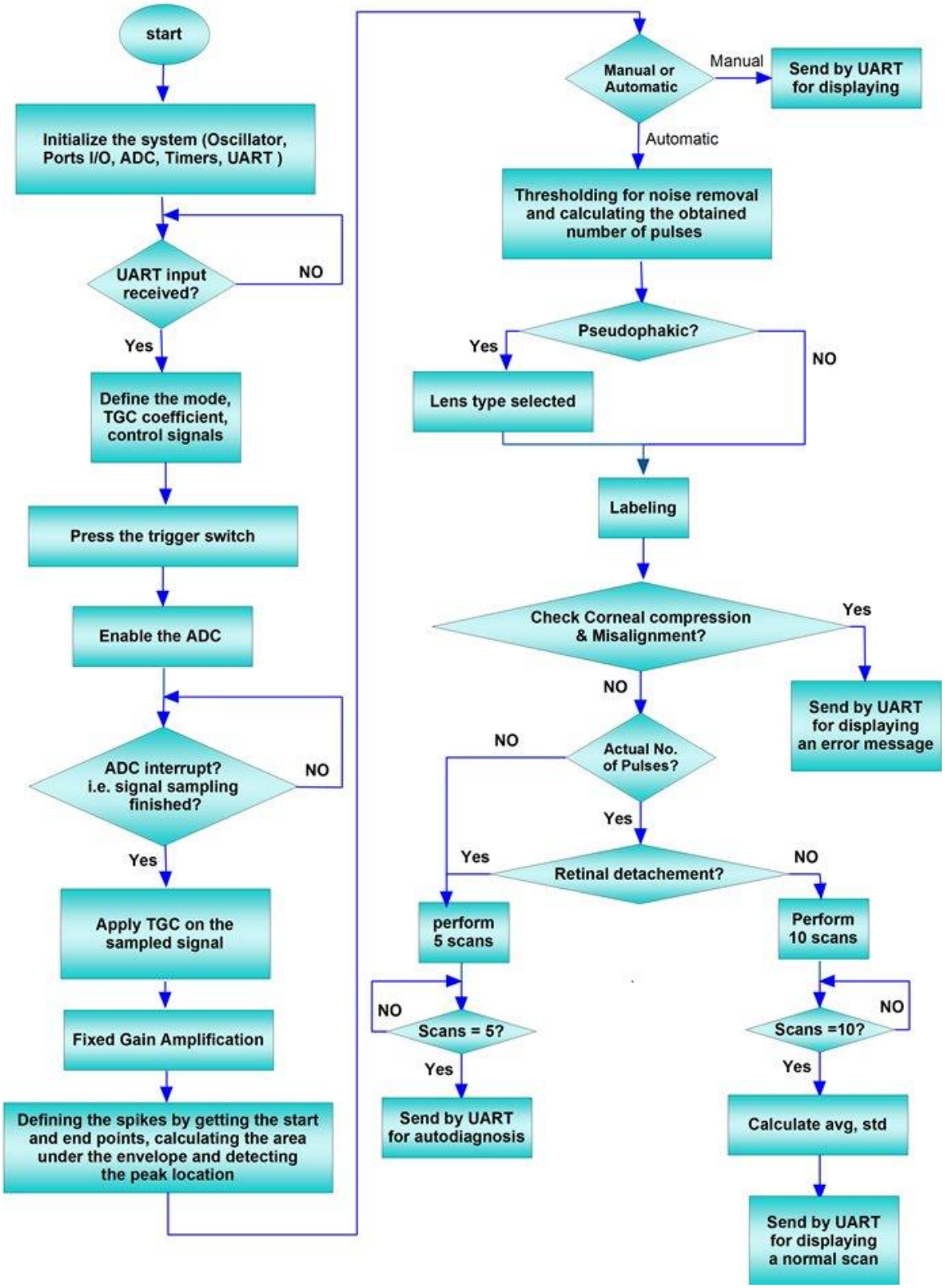


Figure 7.5-1 Firmware Algorithm

7.6 Algorithm Functions

7.6.1Initialization Functions:

These functions have a default set of instruction used to adjust MCU to required state.

```
void init_Clock(void);                  // System clock initialisation
void init_Port(void);                   // General system (ports) initialization
void init_ADC1(void);                   // intialization of ADC
void TIMER3_Init (int counts);          // to set sampling frequency
void init_UART1(void);                  // initialization of UART
void set_Timer1(void);                  // used for setting boud rate of UART
```

7.6.2Interrupt Service Routines

```
void ADC1_ISR (void);                                   //ADC ISR
```

Action: Collects a predefined number of samples in array Fullscan and Raises a flag finished sampling when done

```
void UART1_ISR (void);                                  //UART ISR
```

Action: Collects a fixed number of bytes in temparray then calls Definecommand function to identify what was received by UART.

7.6.3Main Operating Functions

```
void trigger(void);
```

Action:1. Controls TR switch On/Off switching
Sends trigger to Pulser circuit to trigger probe
Enables ADC and its interrupt

```
void TGC(unsigned char *fullscan, const unsigned char a,const unsigned char b,const unsigned char c,const unsigned char d,unsigned int *amplified1);
```

I/P: *fullscan* array of aquired samples per scan
TGC coiffeincents *a,b,c & d* received through UART

Action: Compensate attenuation due to depth reached by Ultrasound beam by increasing gain in a linear fashion as calculated earlier in Chapter 4.
The *fullscan* array is divided into 3 regions.
Region 1 is of fixed gain =c
Region 2 is of linear gain = (d-c)/(a-b)
Region 3 is of gain =d

O/P: *amplified1* array of Time Gain compensated signal

```
void FIXED(unsigned int *amplified1,unsigned int *amplified2,char dcataract);
```

I/P: *ampified1* array from TGC
Dcataract option used to increase gain in case of dense cataract observed by user and received through UART
Action: Amplifies whole signal by a fixed gain of 10 and in case of dense cataract of 15
O/P: am*plified2* array of fully amplified signal

void getpeaks(unsigned int *amplified2 ,unsigned int **allpeakstime,unsigned int **allpeaksamp);
I/P: *ampified*2 array from **FIXED**
Action: Performs Peak detection by:
Definig peak position at index of max amplitude; by calling **peakdetection** and **max** functions.
Defining peak amplitude by calculation area under the envelope (divided to multiple trapeziums) by calling **Peakvalue** and **Trapz** functions.
O/P: 2 arrays one for peak position and one for peak value which dynamically varry by number of detected peaks.
Allpeakstime & Allpeaksamp

void threshold(unsigned int *allpeaksamp,unsigned int *allpeakstime,unsigned int **finalpeaks,unsigned int **finaltime);
I/P: *allpeaksamp & allpeakstime* from **Getpeaks**
Action: Eliminates Peaks under a certain fraction of the initial peak; removes noise from signal.
O/P: 2 arrays one for filtered peaks one for peak position and one for peak value *finalpeaks & finaltime*

void Check (unsigned int *finalpeaks,unsigned int *finaltime,char actualpno,char lenstype,bit pseudo);
I/P: *finalpeaks & finaltime* from **Threshold**
Boolean indicating Pseduophakic eye type is selescted and a byte representing lens type
Action: Includes nested functions to perform all module actions state earlier.
O/P: No direct output is returned to Main module; instead, it either sends data through UART or returns back to automatically aquire a new scan.

void naming(char actualpno,unsigned int mintime[],unsigned int maxtime[],char actualnames[],unsigned int *finaltime, char *names,unsigned int *finalpeaks);
I/P: actual peak number *actualpno* from user through UART
Arrays to define ranges at which each peak is to be found *mintime[] & maxtime[]*
Array of the fixed labels for main peaks in a valid scan *actualnames[]*
Action: In normal case where peak number is coinciding with selected eye type; peaks take the actual names directly.
In case of undeifind peaks, they take the label "x" to be sent to Software for autodiagnosis.
In case of 2 peaks detected in same range; auto labeling cannot be performed and thus would be sent to Software through UART for manual labeling of peaks.
O/P: *names* array with a label for each peak.

bit checklensper(char actualpno,unsigned int *finalpeaks);
I/P: array of peaks amplitude *finalpeaks* and actual peak number *actualpno*
Action: performs misalginemt check by comparing amplitude of 2 lens peaks to the anterior cornea peak.

O/P: returns bit *p1* indicating misalignment fasle or true

bit checkretper(unsigned int *finalpeaks,char actualpno,char *names);

I/P: array of peaks amplitude *finalpeaks* and actual peak number *actualpno*
Action: performs misalginemt check by comparing retina peak to the anterior cornea peak.
O/P: returns bit *p2* indicating misalignment fasle or true

void Avg (unsigned int *ftime[],unsigned int *fpeaks[],unsigned int *Avgtime,unsigned int *Avgpeaks);

I/P: two 2-dimentional arrays including times and amplitudes of 10 valid peaks *ftime & fpeaks*
Action: calculates average postion and amplitude of detected peaks.
O/P: arrays of averge time and amplitude *Avgtime & Avgpeaks*

void Std (unsigned int *fpeaks[],unsigned int *Avgpeaks,unsigned int *StdDev);

I/P: one 2-dimentional array including amplitudes of 10 valid peaks *fpeaks,* array of average amplitudes of each peak *Avgpeaks* from **Avg** function.
Action: calculates standard diviation of amplitude of detected peaks.
O/P: array of standard diviation *StdDev*

void UARTnorm(unsigned int *Avgtime,unsigned int *Avgpeaks,unsigned int* StdDev,char *names);

I/P: Array for peaks time *Avgtime,* for peaks amplitude *Avgpeaks,* for peaks labels *names* and Standard diviaion *StdDev*
Action: Prepares Header and Status bytes for normal peaks and call **UARTloop** function to transfere specifc array.
O/P: Internal calls to **UARTloop** function

void UARTauto(unsigned int *finalpeaks, unsigned int *finaltime,char *names);

I/P: Array for peaks time *finaltime,* for peaks amplitude *finalpeaks,* for peaks labels *names*
Action: Prepares Header and Status bytes for autodiagnosis peaks and call **UARTloop** function to transfere specifc array.
O/P: Internal calls to **UARTloop** function

void UARTloop(unsigned int *tosend,unsigned char *UARTtemp,char noofpeaks)

I/P: current array to be sent through UART *tosend* and number of pulses *noofpeaks*
Action: Divides integer values for time and amplitude to byte and fill them in *UARTtemp* array and transfer through UART.
O/P: Array sent by UART

7.7 Main Data Storage Variables

Due to multiple arrays exchanged through the functions and modules of the project; a summery table of all storage arrays with type and size was required to keep track of when and where to dynamically allocate the array and when to free its memory.

Type	Name	From	To	Size
Main.c				
Unsigned char	*fullscan	ADC	TGC	SamplesPerScan
Unsigned int	*amplified1	TGC	FIXED	SamplesPerScan
Unsigned int	*amplified2	FIXED	Getpeaks	SamplesPerScan
Unsigned int	*allpeaksamp	Getpeaks (peakvalue)	Thresholding	Numberofdpeaks
Unsigned int	*allpeakstime	Getpeaks (peakdetection)	Thresholding	Numberofdpeaks
Unsigned int	*finalpeaks	Thresholding	Check	pulseno
Unsigned int	*finaltime	Thresholding	Check	pulseno
Unsigned char	*UARTtempm	manual UART	UARTloop	(numberofdpeaks*2)+3)
Classification.c				
Unsigned int	*stenv	getpeaks	peakdetection&peakamp	numberofdpeaks
Unsigned int	*endenv	getpeaks	peakdetection&peakamp	Numberofdpeaks
CheckForNormality.c				
char	*names	naming	Check	pulseno
Unsigned int	*ftime[10]	Check	Avg	Pulseno * 10
Unsigned int	*fpeaks[10	Check	Avg	Pulseno * 10
Unsigned int	*Avgtime;	Avg	Check	pulseno
Unsigned int	*Avgpeaks;	Avg	Check	pulseno
Unsigned int	*StdDev;	Std	Check	pulseno
Unsigned char	*UARTtempl	manual UART	UARTloop	((Pulseno * 2) +3)
Serial.c				
Unsigned char	*UARTtemp	manual UART	UARTloop	((Pulseno * 2) +3)

Table 7.7 – 1 Summary of storage arrays

7.8 Main Controlling Flags

Finishedsampling	Indicates ADC finished collecting one scan's samples
Startpacket	Indicates Header Byte revieved by UART
Initrecv	Indicates User Initial Data was received by UART
TGCrecv	Indicates TGC coeifficents received by UART
TGCinterrupt	Indicates editing of TGC has occured
Freezeorcont	Freezes scanning during manual mode
Stop	Stops scanning
Reset	Resets system

7.9 Debugging

During programming phase, multiple errors and deviations occurred. Debugging using logic analyzer, breakpoints and watch window helped fix those errors and ensure that the code performed exactly as requested.

At this level of testing, the target was to ensure that the code is error free and that transitions between modules is guaranteed. Also, various condition controlling looping or exiting the system were manually altered through the watch window and its effect observed using breakpoints.

The second level of testing was to include a simulation signal and check performance specific to the inserted values. This validation stage is to be discussed in **Chapter 9**.

UART data exchange required an even higher level for performance testing that is to use the USB-Serial converter with junction to the Software, this is also discussed in **Chapter 9.**

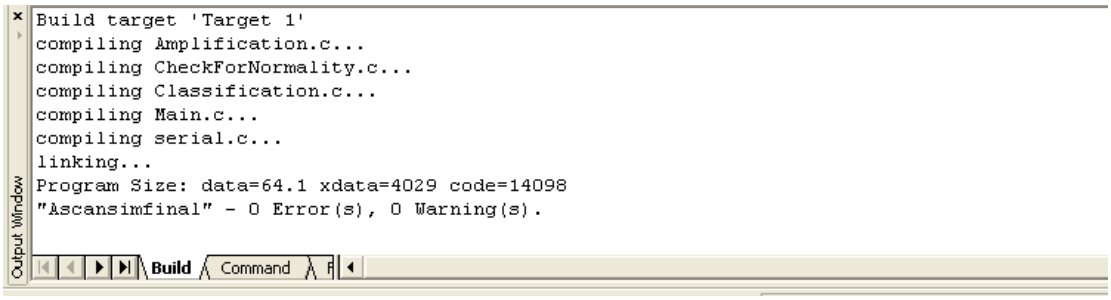


Figure 7.9-1 Debug window; NO Errors and No Warnings

7.10 Performance Analysis

With 25 MIPS microcontroller core, our algorithm achieves sufficient speed coinciding with system real-time acquisition requirement. Further processing is performed post-full acquisition of a single scan. The time required to acquire all 10 scams should be in range of time low enough to ensure accurate, consistent signal acquisition and comfortable probe positioning by User.

7.10.1 Time Analysis

Code Segment	Total # of Instructions	# of Calls	msec	Percentage per Scan Time
Initializations	64	1	0.087	0.2%
Trigger	8	1	--	~0%
ADC Interrupt Service Routine	51	260	0.916	4.0%
UART Interrupt Service Routine	128	--	--	--
UART Transmission	755	1	0.416	1.8%
Amplification (FIXED)	78	1	2.311	10.1%
Amplification (TGC)	385	1	12.570	54.7%
Peak Detection	1037	1	3.784	16.4%
Thresholding	369	1	0. 267	1.2%
Check for normality	2602	1	0. 129	1.2%
Main Loop	591	1	0. 170	1.5%
Mathematical Calculations	307	--	--	--
Dynamic Allocation	320	70	1.451	6.7%
Total Time/ scan	6694	--	26.70	100%
Total Time/ 10 scans	--	--	251.76	--

Table 7.10 -1 Time Analysis of MCU code

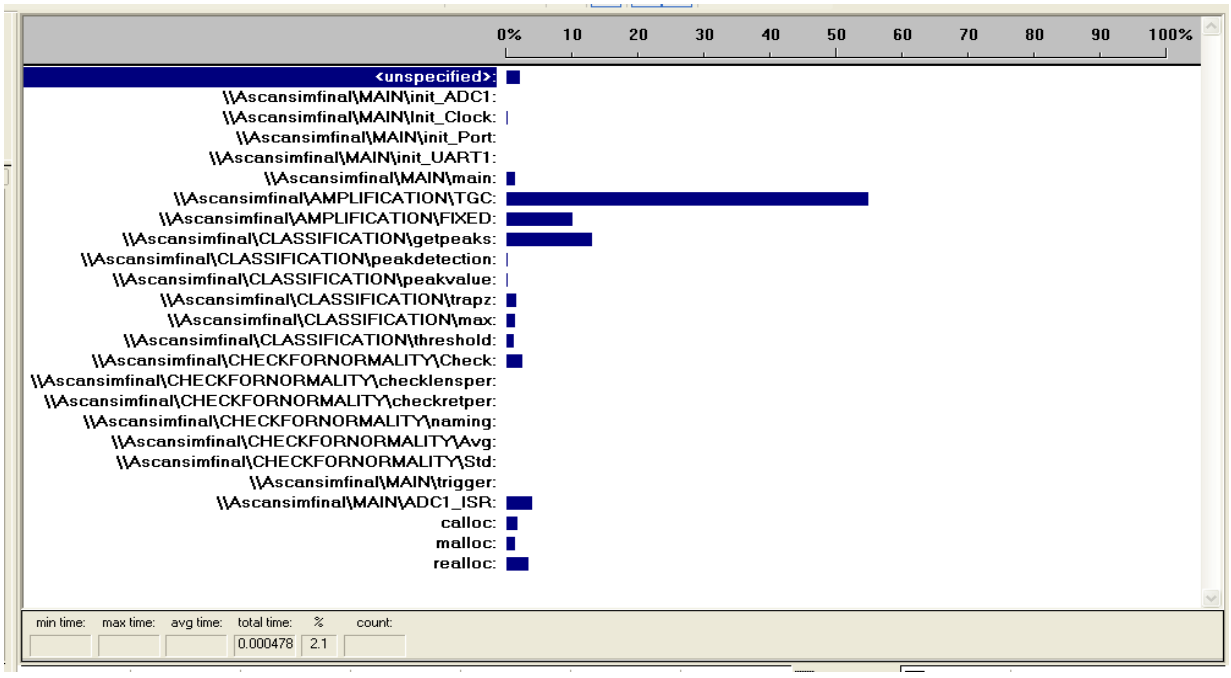


Figure 7.10-1

7.10.2 Memory Distribution:

Code Space: 14138 Bytes
RAM space
Internal: 65 Bytes
External: 3509 Bytes

7.11 Coding Guidelines & Pitfalls

This section is a summery of the obstacles, errors and problems we were subjected to during programming and required research and multiple resource reading to solve.
It's goal us to act as a supporting guide for later programming or for other programmers to not repeat the same errors.

7.11.1 Linking Project Files

1. For multiple .c file project, put all function prototypes in one header file and include in all .c files.
2. Use Header files guard #ifndef Ascan_H and #define directives to prevent redeifintion of header files each time it is included.
3. Differentiate between local header files and complier library header files by inclosing file name in "" or <> with include directive

eg. #include "Ascan.h":	Local File in Prject Directory
#include <stdio.h>	Public Library Header file

4. Must use same memory model in all .c files (1st line of each file) or else will give a warning of incompatible memory model

7.11.2 Variables

1. For Global constants that are not changed and used in multiple files, use #define directive in the header file that is to be included in all .c files.
2. For Global variables that are used in multiple files but are changed in different code segments use extern instead of sending to the function as a pointer.
   Note: Extern Variables are defined in 1 file and when externed must be same data type and memory model
   Eg. Char data pulseno → extern char data pulseno
3. sbit data type cannot be externed, can be defined in the Kit header file c8051f020.h which is also included in all .c files. But, make sure to include this local header file not the public one in the compler library.
4. Ensure exact definition of variables in functions and other files as it affects value stored in variable.
   Eg. Do not exchange a char for unsigned char

5. All flags or control signals are to be defined as bit. Variables that act as counters are defined to char or int according to max value they can hold
6. Do not use float data type as it acquired a huge space, instead use integer and perform scaling of results to reach required accuracy.
7. To send an integer variable through UART must convert to bytes. Use Union which defines a region in a memory in two different ways

eg. union{

int in;

char c[2];

}i2c;

// This definition defines the variable i2c as both an int and a char array of size 2. To access it as int use **i2c.i** and to access it as char use **i2c.c[0]** or **i2c.c[1]**

7.11.3 Functions

1. Variables delcatered inside a function are lost after and cannot be return or seen outside unless they were sent as pointers.
2. When sending a pointer in a function, only the pointer name is sent while the function prototype is defined as apointer.

eg. Call: FIXED(amplified1,amplified2,dcataract);

Prototype: void FIXED(unsigned int *amplified1,unsigned int *amplified2,char dcataract)

7.11.4 Pointers

1. Use pointers when needing a variable sized array and when wanting a function to return more than one variable.
2. Any pointer needs to be initialized before being used.
   Initialization is either to

a) static variable indicating that the pointer points to it

eg. Int a

int*b

or b) to a dynamically located space in the memory during run time.

3. Dynamic Allocation:
    a. Uses malloc (reserve region in memory but does not initialize)
       Calloc (reserves region in memory but initializes to zero)
       Realloc (reallocates region pointed at by a pointer to another)

```
eg.     Int * b

        b= (int*) calloc(size of array, sizeof (int))
```

    b. Requres reserving a region of the memory called memory pool where all the dynamic allocated pointers will be defined. To reserve memory pool use **init_mempool** (address of start of memory pool, sizeof mempool) function.

```
eg.     unsigned char xdata calloc_mempool [0x0DAC]_at_ 0x0000;

                                        // 3500B memory pool

        init_mempool (&calloc_mempool, sizeof(calloc_mempool));
```

    c. When pointer is not needed anymore in the current run of the program, free function is called to free the space it reserved in the memory to be used by other pointers
    d. Make sure never to free a pointer without finishing with it
    e. Never to free a pointer twice
    f. Never to leave pointers hanging without being freed or else will eat up the memory (memory leak)
    g. If pointer is being reallocated inside a loop, first time use calloc then use realloc (donot use realloc directly)
4. Array of pointers is defined as ***ftime[10]** where this is an array of 10 pointers.
   Each pointer is to be allocated independently using a for loop and also freed inddependently.
5. Pointer to a pointer **b is used when a pointer is to be allocated inside a function and its contents are to be seen outside the function(as previously mentioned, variables declared inside a function are lost after its execution)

```
eg. // Final peaks will be allocated inside threshold function and needs to maintain that in
main.
unsigned int **tempfinaltime,*finaltime      //Define pointer and pointer to pointer
Void main(…….)
tempfinaltime=&finaltime;                    // Let pointer to pointer point to pointer
                                             // address
threshold(…,…,…,tempfinaltime);              //Call function sending pointer to
                                             //pointer
finaltime=*tempfinaltime;     //When return let pointer point to the allocated pointer
free(*tempallpeakstime);                     //Free only one of the 2 pointers
void Threshold(…. , ….. , ….. , unsigned int **tempfinaltime)
*tempfinaltime=(unsigned int*)calloc((pulseno+1),sizeof(unsigned int));
//Allocating pointer inside function which affect the original one in main
```

7.12 References

[1] www.cplusplus.com
[2] www.keil.com
[3] https://community.ti.com
[4] http://www.cprogramming.com
[5] http://www.embedded.com
[6] http://msdn.microsoft.com
[7] https://www.ibm.com
[8] Application Notes, SiliconLabs Application note an219 Using Microcontrollers in Digital Signal Processing Applications
[9] Richard G. Lyons , Understanding Digital Signal Processing, Prentice Hall, 2001

Chapter

8

Interfacing

INTRODUCTION
LOW LEVEL PROTOCOL
C8051F020 Micro-Controller
FT232BQ
USB
Verification

HIGHJ LEVEL PROTOCOL
Java to MCU Protocol
MCU to Java Protocol
References

8.1 Introduction

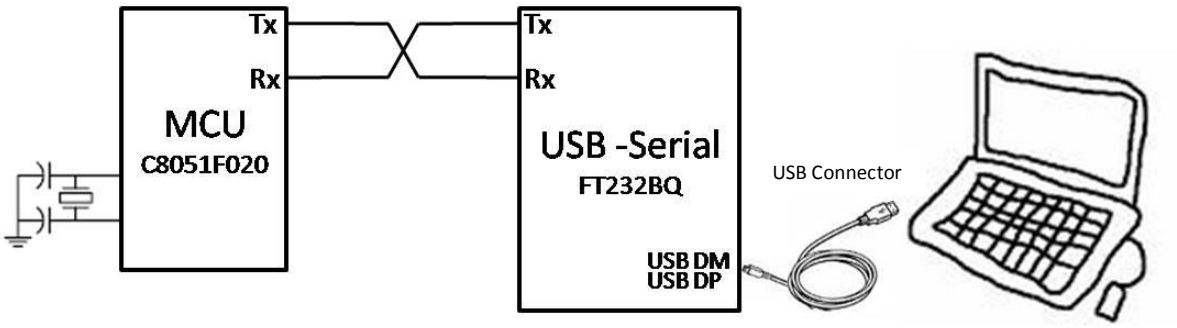


Figure 8.1-1 Overall Block diagram

Figure 8.1 depicts the Interface between the acquired signal and the Laptop. The digitized returning echo is transmitted serially after being processed by the micro-controller. This signal is conveyed over an intermediate converting stage and enters the USB port of the laptop. This chapter will discribe the low level and hogh level protocols that are implemented in order to achieve this data transmission.

8.2 Low Level Protocol

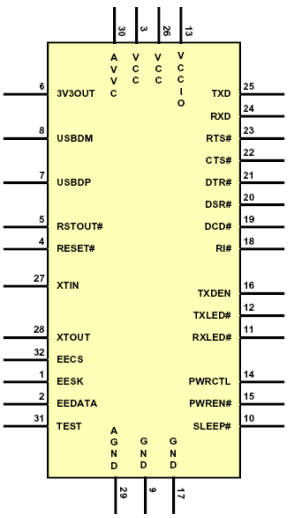
The Low Level Protocol describes the hardware connections between the sub-systems. Pure signal flow is addressed in this section, irrespective of data type and amount. The role of each interfacing component will be explained.

8.2.1 C8051F020 Micro-Controller

Serial Communication is provided from the micro-controller unit over one of its UARTs. UART1 of the micro-controller operates in a full duplex asynchronous mode transmitting and receiving data on the Tx (P0.2) and Rx (P0.3) pins respectively. Both signals are routed via the Priority Crossbar to the Port pins of Port0.The data has a TTL logic level which complies with the required logic level of the USB-Serial converter IC following the micro-controller. Consequently a TTL to RS232 converter IC like MAX232 is not needed in our design.
An important point to consider, while interfacing the MCU to other components, is that the C8051F020 is a 3.3V micro-controller unit, thus providing a high logic level of 3.3V not 5V.

8.2.2 FT232BQ USB-Serial Converter IC

The FT232BQ is a Single Chip USB ↔ Asynchronous Serial Data Transfer IC played the data transmission role. As mentioned in section 4.4, the many features provided by this chip made it the most suitable for reliable, real time data transfer between the MCU and the Java Platform.



The schematic symbol of the QFN32 Package of the FTDI chip FT232BQ is shown is Figure 8.2. You may refer to Appendix VIII to check out the datasheet and view more details about the FT232BQ.[Refer to Appendix VII]

Figure 8.2-1 Schematic diagram (pin out)

8.2.2.1 Power Configuration

The FT232BQ offers multiple choices of Power Configurations, each suitable for a certain type of application. In case of interfacing MCU based designs to USB the most suitable power configuration is the Self Powered configuration with 3.3V logic drive shown in Figure8.3.[Refer to Appendix VII]

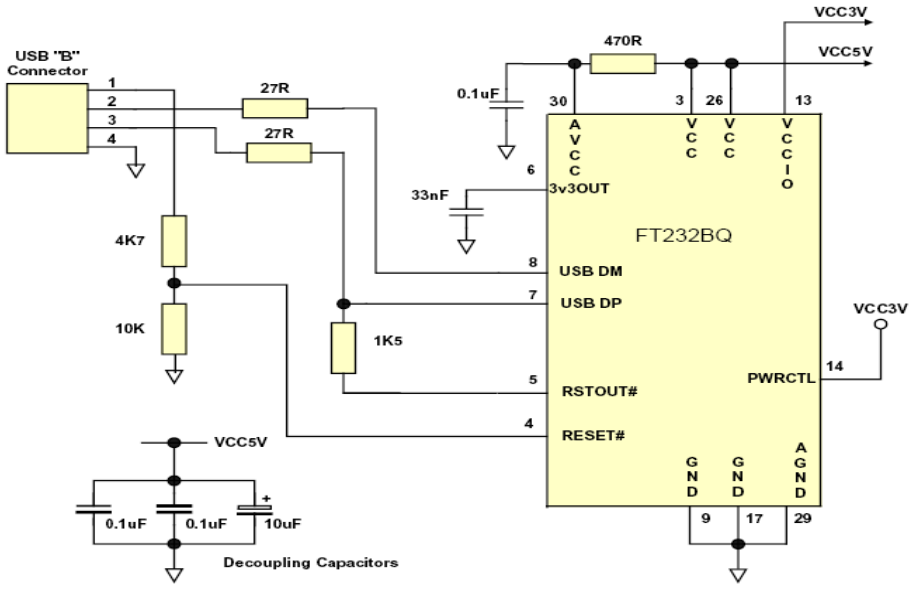


Figure 8.2-2 Power Configuration of FT232BQ

VCCIO is supplied by an external 3.3V supply in order to make the device IO pins drive out at 3.3V logic level, thus allowing it to be connected to a 3.3V MCU or other external logic. We drived the VCCIO pin directly from the MCU Vcc pin to ensure same voltage levels during operation. A USB self powered design uses its own power supplies, and does not draw any of its power from the USB bus. In such cases, no special care need be taken to meet the USB suspend current (0.5 mA) as the device does not get its power from the USB port.

8.2.2.2 USB transfer mode and connection to EEPROM

The FTDI chip FT232BQ features bulk as well as isochronous transfer. While the isochronous transfer mode does not need handshake signals and is programmed by a bit in the external EEPROM, the bulk transfer needs handshake signals but is the default mode of the FT232BQ and needs no external EEPROM.

We considered the usage of the external EEPROM in our design in order to use the isochronous transfer mode, which easier to implement. But upon contacting the FTDI support group (www.ftdichip.com/FTSupport.htm) to ask about the way of programming the EEPROM attached to the FT232BQ we got the following answer:

Hi,
We do not support isochronous transfers in the 232B device, because during the design stage the USB1.1 isochronous specification changed and the chip could not be modified to meet the change.
We will correct our datasheet later.
Thanks
Paul

Paul Huang
Asia Pacific Engineering Manager
Future Technology Devices Intl. Ltd. -
(Taiwan Branch)
Tel: +886-2-87971330
Fax: +886-2-87519737
Email: paul.huang@ftdichip.com
WWW: www.ftdichip.com

Consequently, we had no choice except the bulk transfer mode and employ handshake signals.

8.2.2.3 Handshake signals

The FT232BQ has 2 pins reserved for handshake purposes, RTS (Request To Send) and CTS (Clear To Send). Both pins are active low. Since the C8051F020 MCU does not have dedicated handshaking signals we used general purpose I/O pins of Port 1. There were no threats that the MCU may receive the data with some delay, thus a single wire handshake will do the required work. In this case CTS is tied to ground. When CTS# is active (low) the FT232BQ will transmit any data in it's internal buffers. The FT232BQ drives RTS# high when the available buffer space inside the device drops below 32 bytes. This allows the MCU / logic to continue to send up to 30 characters to the FT232BQ after RTS# goes high without causing buffer over-run. Since we fully control the sending of the MCU and guarantee buffer size sufficiency we don't use the RTS# pin and leave it floating. [1]
The final design of the FT232BQ is shown in Figure 8.4.

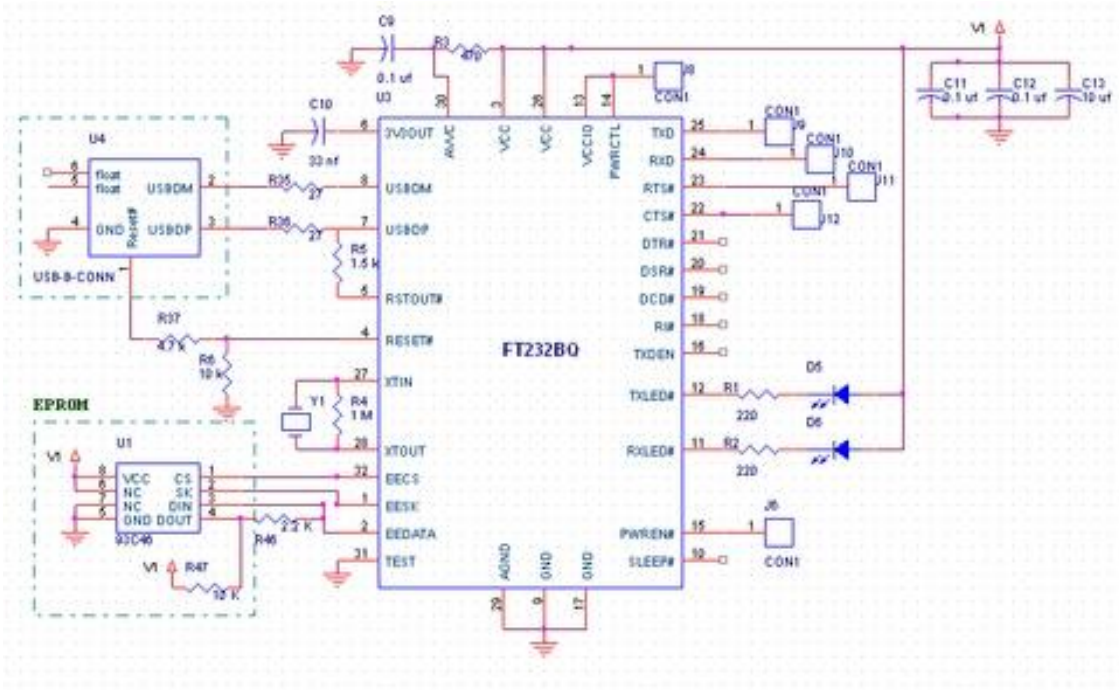


Figure 8.2-3 FT232BQ Connections

8.2.3 USB

The FT232BQ outputs the USB signal on pins USBDM and USBDP. These are connected to a B-type USB connector on the PCB, depicted in Figure 8.4. The Laptop's USB port is connected to the PCB's connector via a USB cable.

8.3 High Level Protocol

In order to exchange data between the microcontroller and the Java PLatform, we built up our own protocol.
Protocol is packet based, with variable number of packets depending on direction of data transmission and the required data to be transferred.
Due to the usage of Bulk transfer in the USB interfacing protocol which guarantees transmission, no checksum was implemented in our protocol.

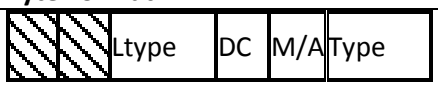
8.3.1 Java –to- Micro Protocol

8.3.1.1 Sites of Usage:

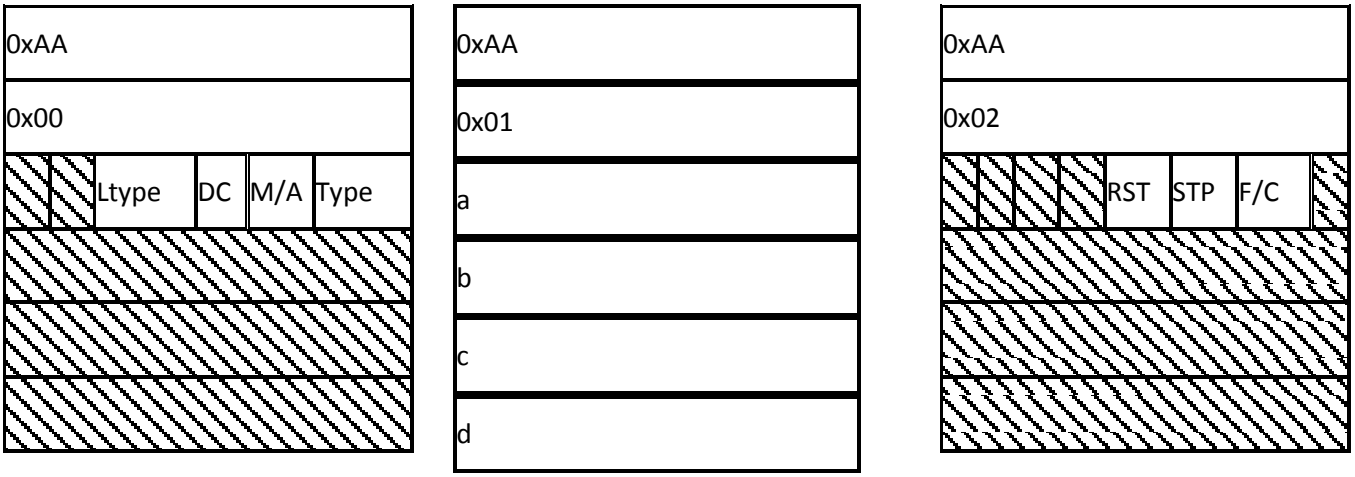
1. Sending initial user data used to control processes in MCU
2. Sending TGC gain coefficients to amplify signal
3. Sending control signals as freeze, continue, stop or Reset.

8.3.1.2 Packet Format

- **# Of Bytes:** A fixed number of 6 bytes is used in three sites of usage
- **Header Byte**: Is used to ensure that data is received from the Software: 0xAA
- **Status Byte:** Used to differentiate the data bytes of the three sites of usage
  0x00: Initial User Data
  0x01: TGC coefficients
  0x02: Control Signals
- **Data Bytes:**

Usage Site	# of Data Bytes	Byte Format
Initial Data	1	 DC: Dense Cataract : 1 M/A: Manual mode : 1 Auto mode : 0 Type: Phakic : 00 Aphakic : 01 Aphakic no lens : 10 Pseudophakic : 11 Ltype(in case of pseudophakic only: PMMA : 00 Acrylic : 01 Silicon : 10
TGC	4	Byte 1: a Byte 2: b Byte 3: c Byte 4: d
Control Signals	1	

		F/C: Freeze : 1 Continue : 0 STP: Stop :1 RST: Reset :1



8.3.2 Micro – to - Java Protocol

8.3.2.1Sites of Usage

1. Sending data for manual mode
2. Sending each scan in auto mode
3. Sending average scan in auto mode
4. Sending auto diagnosis data
5. Retinal Detachment
6. Other diseases
7. Sending unlabeled spikes for manual labeling
8. Sending indication of misalignment & corneal compression

8.3.2.2 Packet Format

- **# Of Bytes:** A variable number of bytes are used based on number of detected peaks
- **Header Byte:**
  Auto mode: 0xAA
  Auto diagnosis: 0xAA
  For Manual Labeling: 0xAA
  Manual mode: 0xBB
  Misalignment or Corneal compression: 0xCC
- **# Of Spikes Byte:** A value indicating number of spikes to be sent
  This Byte is omitted when sending Misalignment or Corneal Compression Error Signal.
- **Status Byte:** Used to differentiate the type of data to be sent and the site of usage sent at.
  For Header Byte: 0xAA

0x00	Auto	Time of spikes
0x01		Amplitude of Spikes
0x02		Std of amplitude of spikes*
0x03		Spike Labels
0x10	Auto diagnosis	Time of spikes

0x11		Amplitude of Spikes
0x13		Spike Labels
0x20	Manual Labeling	Time of spikes
0x21		Amplitude of Spikes

*Std: Standard Deviation is only sent after calculation of average of 10 valid scans.

For Header Byte: 0xBB

0x00	Manual	Time of spikes
0x01		Amplitude of Spikes

For Header Byte: 0xCC

0x00	Error Signal	Corneal Compression
0x01		Misalignment

- **Data Bytes:** Number of bytes varies with the data type being sent.

Eg.

Labels: Char: 1 Byte each

Times, Amplitude or Standard Deviation: Integer: 2 Bytes each

Accordingly, the number of Bytes in a char byte array would be:

Number of Spikes + 3

And Number of Bytes in Integer Array would be:

(Number of Spikes x 2) + 3

HEADER
of SPIKES
STATUS
DATA
…
…
…

0xCC
STATUS