SOFTWARE BASED LOGIC ANALYZER AND SIGNAL GENERATOR

LEE YEAN KIAN
WEK 010133

WXES 3181: Projek Ilmiah Tahap Akhir I
Session 2003 / 2004

Supervisor : Mr Mohd Yamani Idna bin Idris
Modulator : Prof Dato Dr Ir Maskhuri Hj Yaacob

SOFTWARE BASED LOGIC ANALYZER AND SIGNAL GENERATOR

LEE YEAN KIAN
WEK 010133

WXES 3181/3182: Projek Ilmiah Tahap Akhir I & II
Session 2003 / 2004

Supervisor : Mr Mohd Yamani Idna bin Idris
Modulator : Prof Dato Dr Ir Mashkuri Hj Yaacob

ABSTRACT

The logic analyzer and signal generator application is a system where user can input and output through the parallel port. For the logic analyzer application, certain pins of parallel port are "on" by user. There will be a output display on computer. For signal generator application, the parallel port pins will be activated by pressing buttons on a user interface. The parallel port will be connected to a logic analyzer hardware. The output will be display on interface of logic analyzer.

There will be additional function of the application. Software based logic analyzer and signal generator have security feature where every user that uses to the software have to login before using the software. If the user login the wrong user identification and password, then an error message will be displayed. User have to re-login again until the correct user identification and password is typed.

The main advantage of the application is that it can be used to test FPGA or other electronic circuits.

ACKNOWLEDGEMENT

The success completion of this project is related to the contributions of many people. I would like to take this opportunity to thank some of the those people here.

First of all, I would like to state my deepest appreciation to Mr Mohd Yamani Idna bin Idris, my supervisor for giving me this opportunity to develop this project. Secondly, I would like to thank him for his constructive advice, generous guidance, encouragement, support, dedication and supervision along the progress of this project. His diligence and kindness in helping me throughout the project is deeply appreciated.

I also would like to acknowledge Professor Dato Dr Ir Mashkuri Hj Yaacob, as the project moderator who contributed comments, suggestions and ideas to further enhance value of this project.

Not forgetting my course mates, friends and family that have been accompanying me all the time in completing this final year project. Thanks for their unlimited support, invaluable advice, guidance supervision and knowledge and experience sharing.

Last but not the least, I would like to take this opportunity to acknowledge the resource support of the FSKTM which has greatly facilitated the process of the project.

TABLE OF CONTENT

LIST OF DIAGRAM

LIST OF TABLE

CHAPTER 1: INTRODUCTION

1.1 PROJECT BACKGROUND

The title of this system is software based logic analyzer and signal generator. This research is a preliminary research since there is only a few research doing on it. The concept is quite new. This application is mainly how to transfer and acquire data from or to Parallel Port.

The logic analyzer is a simple 4 channel PC parallel port logic analyzer with sample rate 50 kHz. The software uses status ports as input, thus using SPP mode is enough. Luckily most of PCs are equipped with such peripheral. The logic analyzer is built using a switch, 2 buffers, a parallel port, an analog to digital converter (ADC) and a computer. Data is input through parallel port by connecting the switch to the ADC pin. The data will be transfer to an ADC. This is to convert the analog data into digital data. One buffer is used as temporary storage of data that is connected between parallel port and an ADC. Another buffer is to control the transmission of signal. The data will be performed inside computer. There are 2 types of data performance, timing and binary format.

The signal generator, on the other hand, is built using a testing device, parallel port and a computer. Unlike logic analyzer, the data is input from internal computer. The signal generator will be connected to a logic analyzer to display signal generated by signal generator. There will be a user interface that let user to activate or deactivate the parallel port pins. Due to the application is built using only 1 computer, the data that input from

internal of computer will only be tested using a logic analyzer, but not on another computer interface. Each time the specific pin is activated through user interface, the parallel port pin will shows high voltage of 5 volts. Otherwise, it will remains 0 volt.

The problems are how to write and read the data to or from parallel port in clock timing. Normally, the topic include sampling rate and how the data acquisition system is capable of accurately reproducing waveforms of the favorite frequency.

The sampling rate of the logic analyzer and signal generator is defined by the ratio of the number of samples acquired to the duration of sampling. Acquiring 100 samples over a duration of 1 second gives a sampling frequency of 100 Hz. Likewise, acquiring 10 samples over a duration of 100 ms is also a sampling frequency of 100 Hz.

1.1.1 LOGIC ANALYZER

Logic analyzer is a hardware or software that has the ability of analyze signal or waveform. Normally this logic analyzer is connected to a signal generator or a source which can generate signal. Logic analyzer generally have a number of channels and limited timing resolution.

The logic analyzer can be configured as timing analyzer which give a waveform display similar to that of an oscilloscope or as state analyzers which display signals in terms of binary or hex numbers, or both at the same time.

There are 2 types of logic analyzer, software and hardware. The software logic analyzer is a software that have ability of performing several types of waveform. A hardware logic analyzer is a useful electronic circuit which is useful in development and debugging, especially where fast logic circuits are involved with lots of signals whose relations have to be verified or examined.

The data are stored in the memory, in the normal binary code; however, the display can be formatted in various different ways. In most logic analyzers, the data displayed are preceded by a line number (related to the trigger word) for the sake of convenience.

The possible display formats are [Simulink Reference, 1994]:

1. Timing format (Timing analysis)
2. Binary or Hex format (State analysis)

1.1.1.1 OPERATION

The operation of the logic analyzer is controlled by a clock signal (the clock together with the data input is the data acquisition unit). A sample of the data present is taken each time a clock pulse occurs, and transferred to the memory [Craig Maynard, 2000].

After the first clock pulse, one data word is stored in the first memory location. On receipt of the next clock pulse this data word is shifted one place further, and the next data word is transferred to the first memory location. Each subsequent clock pulse causes the string of data words to be shifted one place further in this way.

1.1.1.2 ADVANTAGES

- A logic analyzer is useful in electronic development and debugging, especially where fast logic circuits are involved with lots of signals whose relations have to be verified or examined.
- The logic analyzer is used in order to make accurate timing measurements and to investigate voltage vs time characteristics of signals.

1.1.2 SIGNAL GENERATOR

Signal generator, on the other hand, is reverse of logic analyzer. Unlike logic analyzer, signal is generate by signal generator and is transmit out to a logic analyzer through parallel port. The signal generator generate signal that may be used in testing, audio analysis, or recording [Martin Clausen, 2002].

Generally, signal generator is divided into 2 types, hardware and software.

A software signal generator is a software that capable of sending a signal down a cable so that it can be traced by a detector receiving signals on the same frequency.

A hardware signal generator is connect to some DAC (Digital Analog Converter). The number of DAC that is intend to be connected is depends on the number of bits supported by the signal generator. The DAC have the ability to convert digital data into analog data.

There are 2 types of signal that can be generated by signal generator, digital and analog. Digital signal is data in its original format, it is binary. However signal generator can

convert the binary format into hexadecimal as well. Analog signal capable to be generated using the DAC device.

1.1.2.1 OPERATION

Signal generator generate data through 2 process, digital frequency generation and analog processing of signal [Mautin, 2001].

- **Digital Frequency Generation**

  The digital part of the frequency generator consists of a shift register, an adder, a latch and a 1M bit EPROM. The shift register reduces the need of port pins at the MCU from 33 to 3. It also synchronizes the data input from the MCU with the operation of the adder.

  The adder output is feedback to itself via the latch. Therefore the value at the output of the latch is increased by the value in the shift register at every clock cycle. This value is also taken as an address for the EPROM. This EPROM contains a table, which allows to convert the value from the latch into the amplitude of the output signal. In principle any waveform can be stored and generated.

  The frequency and amplitude modulation is based on a DDS software in the micro controller. Since the sinus is read from a 16K byte lookup table (LSB first) starting at &H01000, we can replace it by any other waveform.

- **Analog Processing of Signal**

The final output signal is formed through DAC, filter, multiplier and an amplifier. The digital output of the EPROM is converted to a analog signal by a high speed, high precision DAC. Its current output is converted to a voltage by a operational amplifier.

The MCU shifts the amplitude value into the DAC. The output of the DAC is feed into the multiplier and allows the adjustment of the amplitude. The offset is adjusted by another DAC. Its output is converted to a bipolar output by a dual OP. In the final step the signal is amplified by a strong operational amplifier.

1.1.2.2 ADVANTAGES

- Used to generate multiple types of signals.
- The concept of signal generator can be used in Smart home. Smart home is a new technology that is currently being accepted and adopted in the society today. The similar concept between smart home and signal generator is the automation of lights. This include turning on and off lights and appliances based on the lifestyles of the homeowner.

1.2 PROJECT OBJECTIVES

- Is used to test FPGA.
- Provide user to input data from external of parallel port into computer.
- Provide user to input data from computer and output to parallel port.
- Verify the input and output of parallel port.

1.3 PROJECT SCOPE

There are 2 parts that will be developed, there are logic analyzer and signal generator [Mautin, 2001].

Logic analyzer

- data acquisition

  logic analyzer must has the capability of reading data from parallel port pins.

- output display

  logic analyzer can displays 2 types of outputs, there are timing and binary. Timing output is square waveform.

Signal generator

- output data

  signal generator must has the capability of output data from parallel port pins. The data output into parallel port can be test using a logic analyzer.

- Input feature

  Signal generator has the feature of input data. Generally, a user interface will be created to let user activates the parallel port pins.

1.4 PROJECT CONSTRAINT

There are a few constraints that need to be considered when implementing the system. There are as below [Jan Axelson, 1999]:

- **Parallel port type**

  Parallel port consists of many types. The available types are SPP, PS/2 type (simple bi-directional), EPP, ECP and multi-mode ports. The most suitable mode for the application is SPP. The reason SPP port is chosen is because SPP has 5 status inputs and 4 bi-directional control port. In newer ports, there are 8 data ports that can be used as inputs.

- **Parallel port pin**

  Every parallel port has three types of pins, data, status and control. The parallel port pin that is selected to read is status pin and data pin is used to write. The data pin consists of 8 status pins, D0-D7 and 4 status pins, S3, S4, S5 and S7.

- **Parallel port speed**

  The speed of parallel port is 200-400 kHz.

- **Input for logic analyzer**

  The input for logic analyzer must consists voltage of 0-5 volt. This is because the parallel port pin can not support power more than 5 volt. Otherwise, the parallel port will break.

- **Sampling rate**

  The sampling rate of logic analyzer and signal generator is 50-100 kHz. The bigger the frequency, the more accurate the logic analyzer and signal generator become.

- **Output of logic analyzer and signal generator**

  Logic analyzer can display data in square waveform, bus and binary format. Signal generator can generate square waveform.

1.5 REPORT LAYOUT

The purpose of this layout is to give an overview of the major phases involved during development of the project. Below is the report layout:

Chapter 1: Introduction

This chapter gives an overview of the major phases of the project that includes the objective, project overview, project scope, project development methodology and project schedule.

Chapter 2: Literature Review

This chapter give brief explanation on topics researched and studies that are relevant to this project. It is the combination between literature search and literature review. Among the discuss topics are operating system and Web development technology. Besides that, this chapter also makes a study on logic analyzer and signal generator.

Chapter 3: System Analysis

This chapter emphasizes on the analysis of the project's requirements, functional requirements, non-functional requirements and the methodology of system. It explains how the requirements for this project were acquired and the results of the analysis.

Chapter 4: System Design

This chapter explains the conceptual and technical design of the system. It covers the Structure chart, content design, block diagram, data flow diagram and user interface design.

1.6 PROJECT SCHEDULE

A project schedule is needed to achieve the project objectives. It was planned to manage the time and tasks needed to accomplish.

For this project, it was planned in a set of several milestones as below:

Table 1.6A: Project schedule

MONTH	JUN				JULY				AUGUST				SEPTEMBER			
WEEK	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Project Definition																
Literature Review																
System Analysis																
System Design																
Documentation																

1.7 SUMMARY

This chapter is basically the introduction of the software based logic analyzer and signal generator application. This chapter has been clearly clarified the project background, objectives, scope, constraint, report layout and schedule of this project as well as the overview of the system. The next chapter will describe some research and analysis of the project.

CHAPTER 2: LITERATURE REVIEW

This chapter will carry out some research and analysis of the project or generally known as literature review. In the process of developing this system, researches have been done to understand various concepts as background study, especially logic analyzer, signal generator, parallel port and parallel port programming. Another important aspect of literature review is to sufficiently equip the developers with the knowledge of the strength and the limitations of several system architecture, system platform and also programming tools before the final decision in selecting the most suitable and appropriate development tools. In this chapter, some further understanding of the project related terms, terminologies and technologies were discussed.

2.1 PURPOSE OF LITERATURE REVIEW

Review of literature is a background study about the knowledge and information gained to develop this project. Its' purpose is to get a better understanding on the development tools that can be used to develop a project and also to get a better knowledge on the development methodologies used while developing a project. Besides that, review of literature also enables the developers to do comparison on the past-developed projects and study the strength and weakness of it. It will also give an overview of how to improve the weakness and fulfill the requirements needed.

2.2 INFORMATION GETHERING METHOD

In developing a system, it is important to identify the system requirement. In order to identify them, a lot of information is needed. A few techniques have been used to find out

what the system needs and users really want. The requirement elicitation takes quite a long time. This is due to several techniques need to be applied in order to get a complete requirement.

2.2.1 FACT FINDING TECHNIQUES

Fact-finding is needed in order to have a better understanding of the system's needs and requirements. There are many sources that provide information in my research. The information gathering techniques involved are:

- **Internet Research**

  Internet is used as the main resource for referring any ambiguities that arise during the entire development period. By analyzing in the similar system has made a big help in giving ideas on the features, functionality as well as the design of the web-based system. Besides that, online tutorials regarding programming language can also be obtained through surfing the Internet.

- **Document Room**

  Previous seniors' thesis have been read through in order to gain an overall understanding on how a system was developed, what were the functional and non-functional requirements, and other related data. The general structure of

each thesis has also been observed to find out the steps taken in carrying out a thesis.

- **Library**

  Books, journals and magazines from the library have been read through and valuable information has been noted down.

- **Bookstores**

  Several renowned bookstores, such as MPH and Popular, contain relevant references about this project. Suitable references books have been purchased in order to learn in detail about the technologies used in this project.

2.2.2 SOFTWARE TESTING

Relevant software and web development tools have been tested out to evaluate their suitability for this project.

2.2.3 OBSERVATION

The current logic analyzer and signal generator application had been reviewed. Through this technique, the related features of those application had been observed and defined. Besides, the related system is found and tested to find out the functionality. Review has been carried out to see whether it is economic to apply the new system and whether there is enough equipment to develop the new system.

2.3 LOGIC ANALYZER

2.3.1 DEFINITION

Logic analyzer is a hardware or software that has the ability of analyze signal or waveform [Arian, 2003]. Normally this logic analyzer is connected to a signal generator or a source. There are 2 types of logic analyzer, software and hardware.

The software logic analyzer creates virtual analyzer on the computer's screen and allows easy manipulation with all controls by mouse. A software logic analyzer is basically nothing more than a memory, which functions here as a multiple shift register for at least 16 bits in parallel, where user store digital data, taken from a system under test. A hardware logic analyzer is a like a recorder for digital signals. During a certain (small) period of time, the state of a few digital lines can be recorded to a file. An event can be specified to signal the start of the recording, i.e. line 1 toggeling from 0 to 1. This recording can be viewed afterwards, allowing for zooming and scrolling in the time domain.

A logic analyzer is useful in electronic development and debugging, especially where fast logic circuits are involved with lots of signals whose relations have to be verified or examined.

2.3.2 MODE

In fact, there are two modes of logic analyzers, timing and state [Agilent Technologies, 2000].

- **Timing analyzer**

A timing analyzer is the part of a logic analyzer that is analogous to an oscilloscope. Timing mode uses an internal user-defined clock for data capture. As a matter of fact, they can be thought of as close cousins. The timing analyzer displays information in the same general form as a scope, with the horizontal axis representing time and the vertical axis as logic levels of high and low. Because the waveforms on both instruments are time-dependent, the displays are said to be in the "time domain". Timing mode displays more information than state mode than is often necessary.

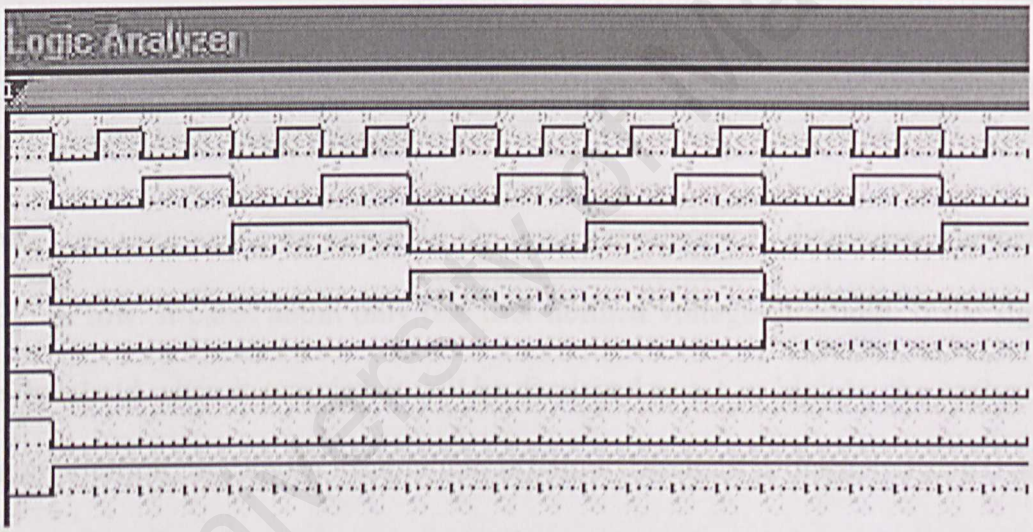


Diagram 2.3.2A: Logic analyzer timing mode [Agilent Technologies, 2000]

- **State analyzer**

A state analyzer is analyzer that performs its data in binary or hexadecimal way. The binary format takes the same binary coded data from the memory as for the time mode, and represents them directly in ones and zeroes to give what is known as a state table. The hexadecimal representation which is simply a

translation of the binary format into the hexadecimal notation (where the numbers from O to 15 are represented by 0, 1,2, 3,4, 5, 6, 7, 8, 9, A, B, C, D, E, and F, successively). For example, D7 in "hex" = binary 1101 0111 = 215 in normal decimal notation.

State mode is generally clocked by the clock in the system that is being tested. It can also be useful to connect the clock read or write pin or the address latch enable pin. This allows data to be captured every time the read or write pin changes, or whenever the address latch is enabled. The state analyzer only captures data when the clock changes state. Depending on how it is configured, the data may be captured on the rising or falling edge of the clock.

2.3.3 SAMPLING METHOD

A timing analyzer works by sampling the input waveforms to determine whether they are high or low. It cares about only one user-defined voltage-threshold. If the signal is above threshold when it samples, it will be displayed as a 1 or high by the analyzer. By the same criterion, any signal sample that is below threshold is displayed as a 0 or low. From these sample points, a list of ones and zeros is generated that represents a one-bit picture of the input waveform. As far as the analyzer is concerned, the waveform is either high or low no intermediate steps. This list is stored in memory and is also used to reconstruct a one-bit picture of the input waveform, as shown below.

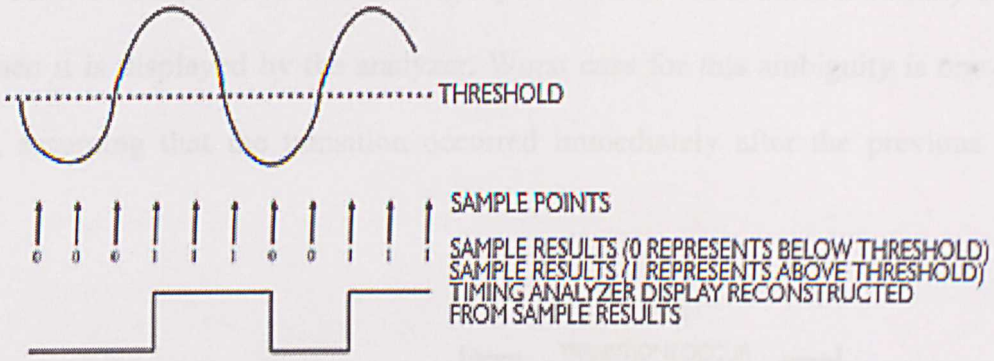


Diagram 2.3.3A: Input waveform [Agilent Technologies, 2000]

This tendency to square everything up would seem to limit the usefulness of a timing analyzer. User should remembers, however, that it is not intended as a parametric instrument. If user wants to check rise time of a signal with an analyzer, user are using the wrong instrument. But if user needs to verify timing relationships among several lines by seeing them all together, a timing analyzer is the logical (no pun intended) choice. For example, imagine that user has dynamic RAM in a system that must be refreshed every 2 ms. To ensure that everything in memory is refreshed within that 2 ms, a counter is used to count up sequentially through all rows of the RAMs and refresh each. If user wants to make certain that the counter does indeed count up through all rows before starting over, a timing analyzer can be set to trigger when the counter starts and display all of the counts. Parametrics are not of great concern here user merely want to check that the counter counts from 1 to N and then starts over.

When the timing analyzer samples an input line, it is either high or low. If the line is at one state (high or low) on one sample and the opposite state on the next sample, the analyzer "knows" that the input signal transitioned at sometime in between the two samples. It doesn't know when, so it places the transition point at the next sample, as

shown below. This presents some ambiguity as to when the transition actually occurred and when it is displayed by the analyzer. Worst case for this ambiguity is one sample period, assuming that the transition occurred immediately after the previous sample point.

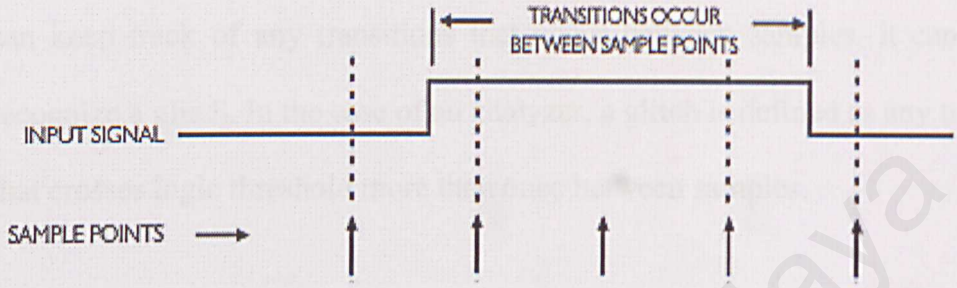


Diagram 2.3.3B: Sample point [Agilent Technologies, 2000]

With this technique, however, there is a trade-off between resolution and total acquisition time. Remember that every sampling point uses one memory location. Thus, the higher the resolution (faster sampling rate), the shorter the acquisition window.

2.3.4 FEATURES

- **Glitch capture**

  One headache of digital systems is the infamous "glitch". Glitches have a nasty habit of showing up at the most inopportune times with the most disastrous results [Craig Maynard, 2000]. How does user captures a glitch that occurs once every 36 hours and sends system into the weeds? Once again the timing analyzer comes to the rescue! Some timing logic analyzers have glitch trigger capability that makes it easy to track down elusive glitch problems.

A glitch can be caused by capacitive coupling between traces, power supply ripples, high instantaneous current demands by several devices, or any number of other events. High performance DSO's provide a glitch trigger mode where the user can specify a trigger on a pulse of less than or greater than a specified pulse width. However, since a timing analyzer samples the incoming data and can keep track of any transitions that occur between samples, it can readily recognize a glitch. In the case of an analyzer, a glitch is defined as any transition that crosses logic threshold more than once between samples.

The analyzer already keeps track of all single transitions that occur between samples. To recognize a glitch, user "teaches" the analyzer to watch for multiple transitions and trigger on them.

It's helpful to have the ability to trigger on a glitch and display data that occurred before it. This can help user to determine what caused it. This capability also enables the analyzer to capture data only when user wants it when the glitch occurred. Think about the example mentioned in the beginning paragraph of this section. User has a system that crashes periodically because a glitch appears on one of the lines. User could set up a trigger condition that captures the crash and then look at the control lines before the crash occurred. Another alternative is to use an analyzer without glitch trigger capability and sit in front of the machine waiting until user see the glitch. Unfortunately, neither of the above are practical alternatives. If user can tell the analyzer to trigger on a glitch, it can stop when it finds one, capturing all the data that happened before

it. User lets the analyzer be the baby-sitter and when the system crashes, user has a record of what led up to the error.

- **Triggering**

Another term that should be familiar to logic analyzers is triggering, often called "trace point." A logic analyzer continuously captures data and stops the acquisition after the trace point is found to display the data. Thus a logic analyzer can show information prior to the trace point, which is known as negative time, as well as information after the trace point.

a)Pattern Trigger

Setting trace specifications on a timing analyzer is a bit different from setting trigger level and slope on an oscilloscope. Many analyzers trigger on a pattern of highs and lows across input lines.

b)Edge Trigger

Edge triggering is a familiar concept to those accustomed to using an oscilloscope. When adjusting the "trigger level" knob on a scope, user could think of it as setting the level of a voltage comparator that tells the scope to trigger when the input voltage crosses that level. A timing analyzer works essentially the same on edge triggering except that the trigger level is preset to logic threshold.

Why include edge triggering in a timing analyzer? While many logic devices are level-dependent, clock and control signals of these devices are often edge-sensitive. Edge triggering allows user to start capturing data as the device is clocked. As a simple example, take the case of an edge-triggered shift register that is not shifting data correctly. Is the problem with the data or the clock edge? In order to check the device, user need to verify the data when it is clocked on the clock edge. The analyzer can be told to capture data when the clock edge occurs (rising or falling) and catch all of the outputs of the shift register. Of course, in this case user would have to delay the trace point to take care of the propagation delay through the shift register.

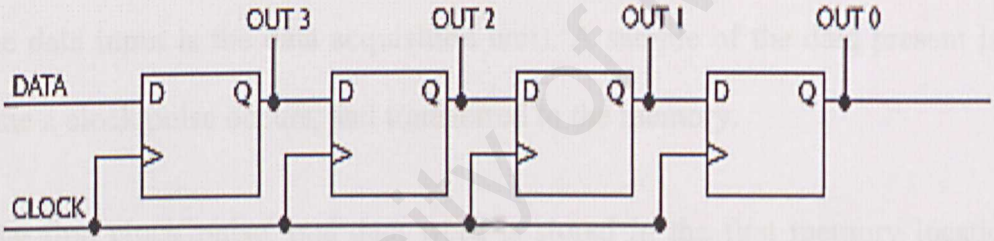


Diagram 2.3.4A: Register [Agilent Technologies, 2000]

Triggering Conditions

Trigger patterns can be defined to tell the logic analyzer when to start capturing data. Any input signal channel can be set to a variety of trigger conditions. Data capture begins when all of the trigger conditions in the active trigger pattern are satisfied. Table 1 lists possible trigger conditions for each channel.

Table 2.3.4B: Channel trigger conditions

Table 1. Channel Trigger Conditions	
Trigger Condition	Description
Don't Care	Default trigger condition. The channel is not used to determine the trigger event.
Low	The analyzer triggers when the channel is low.
High	The analyzer triggers when the channel is high.
Falling	The analyzer triggers when the channel is falling.
Rising	The analyzer triggers when the channel is rising.
Rising or Falling Edge	The analyzer triggers when the channel is rising or falling.

2.3.5 OPERATION

The operation of the logic analyzer is controlled by a clock signal (the clock together with the data input is the data acquisition unit). A sample of the data present is taken each time a clock pulse occurs, and transferred to the memory.

After the first clock pulse, one data word is stored in the first memory location. On receipt of the next clock pulse this data word is shifted one place further, and the next data word is transferred to the first memory location. Each subsequent clock pulse causes the string of data words to be shifted one place further in this way.

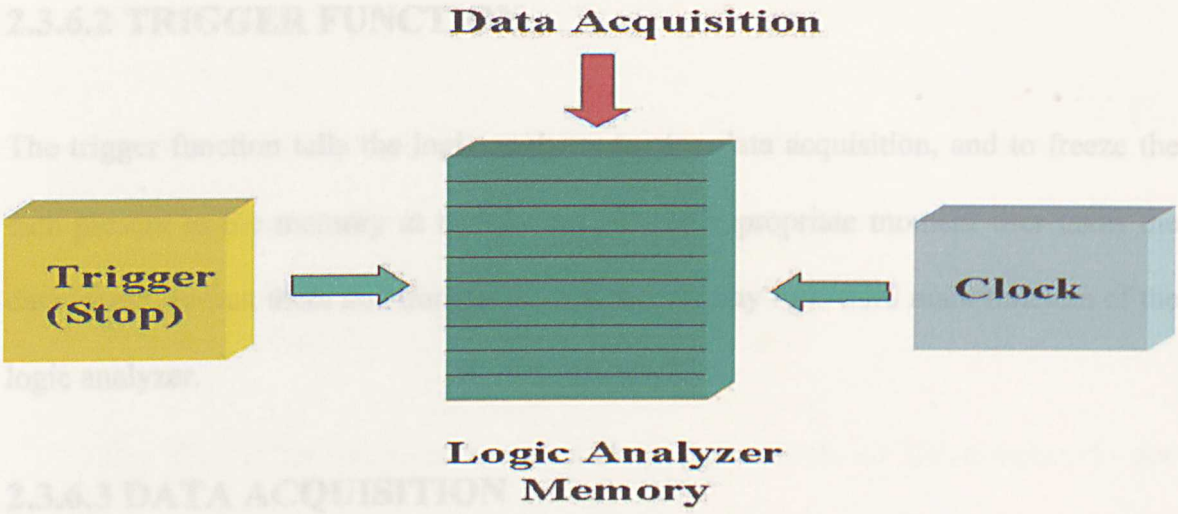


Diagram 2.3.5A: Data acquisition [Craig Maynard, 2000]

2.3.6 COMPONENTS

2.3.6.1 MEMORY

A prime consideration for defining the memory of a logic analyzer is the width of user data word, in other words how many bits in parallel can be taken at every clock pulse. The number of words user can store at a time is a second consideration. It may be e.g. 64; the memory will then have 64 locations for the data words – but any other size is possible.

Now in order to visualize the data of interest, user need to be able to freeze the data flow at a required spot. This is done with the aid of the trigger function - a very important part of every logic analyzer.

2.3.6.2 TRIGGER FUNCTION

The trigger function tells the logic analyzer to stop data acquisition, and to freeze the data present in the memory at that instant. At the appropriate moment user takes the data stored, format them and transfer them to the display - the third main function of the logic analyzer.

2.3.6.3 DATA ACQUISITION

The sampled data should, of course, be acquired correctly from the system under test. That it is done via probes, color-coded leads being used to connect the probes to the system under test.

Depending on the type of analyzer, user can also use individual probes for connecting each data input to the circuit of interest, for example for sampling the data present on the data bus and address bus of a microcomputer system as shown in diagram 2.3.6.3A.

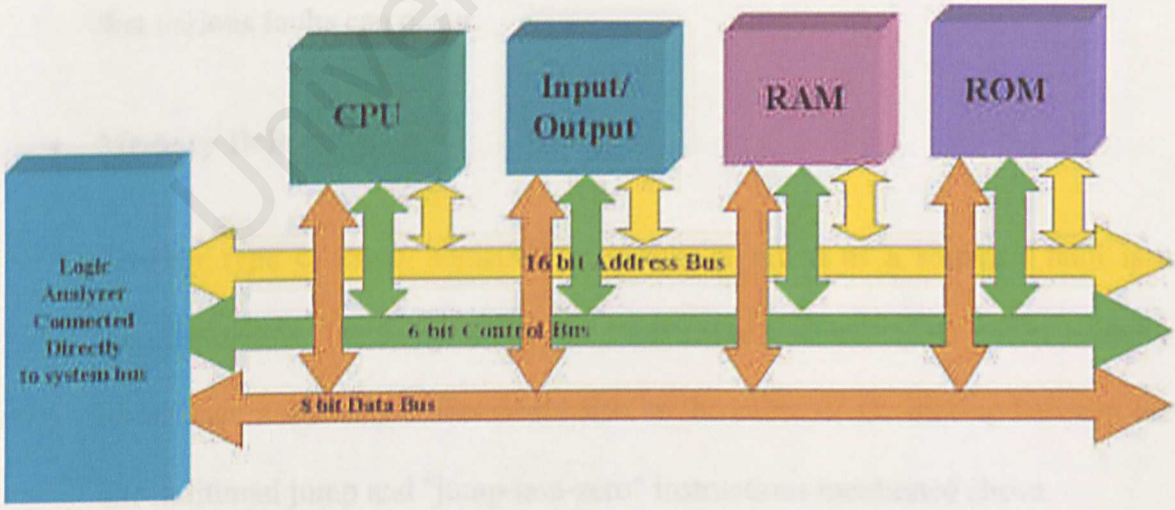


Diagram 2.3.6.3A: Transmission of data through bus [Craig Maynard, 2000]

2.3.7 SOFTWARE FAULTS

Software errors are mainly program errors [Craig Maynard, 2000]. These can be subdivided into 3 main groups:

- **Wrong Instructions**

  The first is an error caused by wrong instructions, so the program is not executed in the way planned. This could also be due to missing instructions, wrong addresses, etc.

- **Timing Faults**

  In some programs for communication between processor and peripherals, the transmission rate is software, controlled. If the peripheral unit is too slow in executing its function and the software is not programmed to wait for completion of the operation in question, the program will continue, which means that various faults can occur.

- **Memory Defects**

  Another type of error, sometimes wrongly regarded as a software fault is a memory defect - due e.g. to an open circuit or a short-circuit in the memory lines. Such a memory defect could also be the cause of the mix-up between the unconditional jump and "jump-non-zero" instructions mentioned above.

2.3.8 HARDWARE FAULTS

The faults which can occur in digital circuitry may be due to the hardware, the software, or both; software faults occur, of course, only if the application is controlled by a software program [Craig Maynard, 2000]. The following hardware faults may be distinguished:

- no data or wrong data
- glitches
- spikes
- races
- timing error
- ringing
- wrong level

These faults can also occur in analog circuits; however, the large number of Signal lines in combinational circuits can greatly complicate the task of fault finding.

Software faults are mainly program errors from different sources, or memory defects. Let us now look at each type of fault in turn.

- **No data**

  No data or wrong data can be due to a broken line, short-circuit, or power failure in some circuits.

- **Glitches**

Diagram 2.3.8.2A shows one way in which a glitch can be generated. The circuit is a 4-bit counter gated in such a way that it counts only up to 3. At count 3 both outputs a and b are high; the AND gate is then enabled and both flip-flops are reset to zero. This means that the "a" output is high only for a very short time here before it is reset. The resulting short pulse (shown in the figure) is called a "glitch". This type of unwanted signal will normally be found only in the design stage. Correct design should eliminate all glitches.

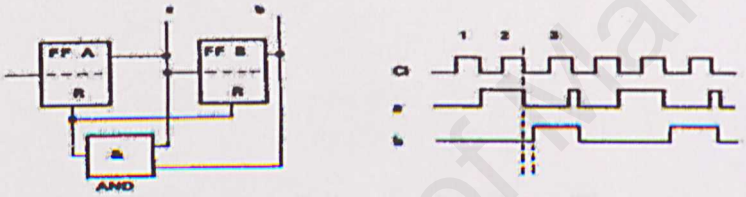


Diagram 2.3.8A: Glitch generation [Craig Maynard, 2000]

- **Spikes**

A spike is an unwanted signal very similar to a glitch. Diagram 2.3.8B shows an example of spike generation. The sharp rising edge of a signal transition present on one or more of the lines in a bus system can cause a parasitic signal to appear on one or more bus lines owing to capacitive coupling.

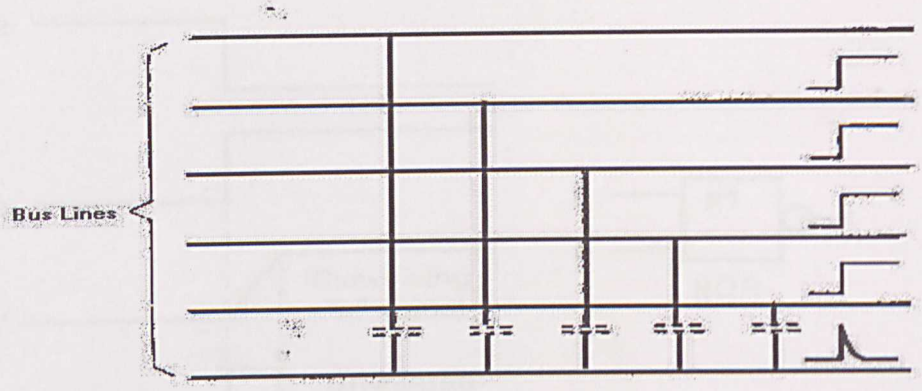


Diagram 2.3.8B: Spikes [Craig Maynard, 2000]

This parasitic signal is called a spike. It can create problems in a logic circuit connected to the line involved as the circuit would receive the unwanted signal for a short time.

- **Races**

  Races are quite similar to glitches and spikes. They generally occur when signals of different speed are combined in one logic circuit. See diagram 2.3.8C. If both signals a and b have sharp edges as shown in the top half of the figure there will be no signal at the output of the NOR gate. However heavy loading of the "b" signal line can make the rising edge of the "b" signal somewhat slower than the falling edge of the "a" signal. Consequently, it will take longer than usual for b input to reach its switching threshold. For a short time, therefore, both a and b will be within the range characterized as "low", giving a high output.

  As soon as the b signal reaches the threshold level, the output goes low which results in a short unwanted output pulse.

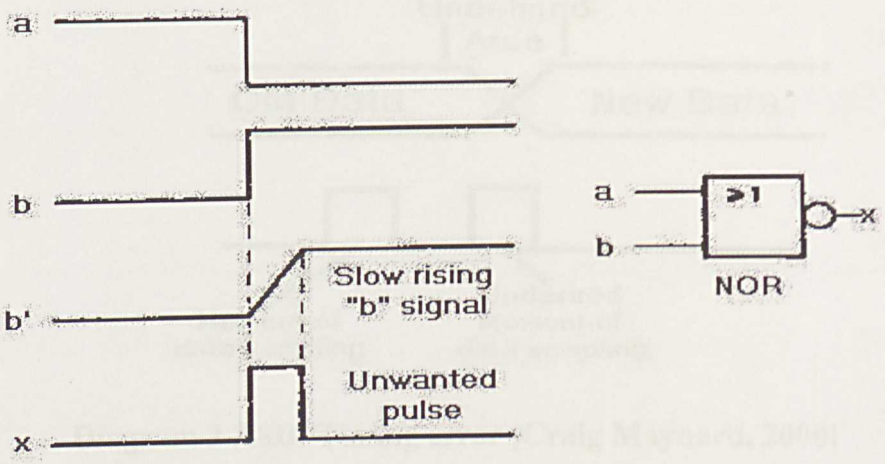


Diagram 2.3.8C: Races [Craig Maynard, 2000]

- **Timing Errors**

Another hardware fault condition can be caused by timing errors.

Diagram 2.3.8D shows data on a data bus together with the corresponding sampling pulse. Let suppose the data is taken from the bus and sent to another device at the falling edge of the sampling pulse. If, for some reason the data sampling pulse is delayed and appears at the moment when the data are changing (indicated by the crossing of the data lines), undefined data are sampled and the result may be false.

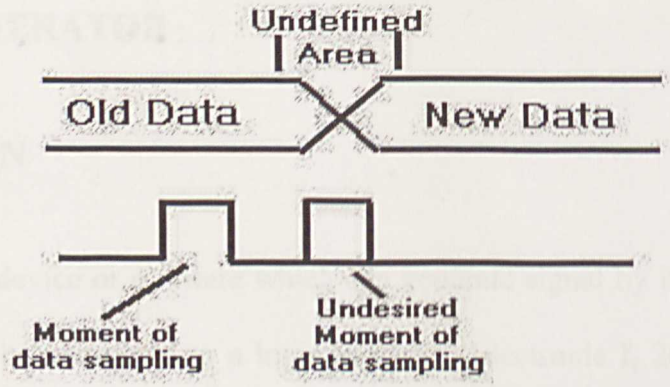


Diagram 2.3.8D: Timing error [Craig Maynard, 2000]

- **Wrong Level**

Each family of logic circuits has its own defined range of switching levels.

A TTL output will deliver an output level of less than 400 mV when the signal is logic 0, and of at least 2.4 V when the input signal is logic 1. At the input, however, it will still recognize 800 mV as a O and signals from 2 V onwards as 1. If the input level is between 0.8 and 2.0 V, the logic condition is undefined. The output signal can then be either a 1 or O, so a faulty condition can occur. These wrong levels can be caused by excessive loading of the logic gates or by interference.

2.4 SIGNAL GENERATOR

2.4.1 DEFINITION

Signal generator is a device or software which can generate signal by itself. Normally, signal generator will be connected to a logic analyzer [Electronic I, 2003]. There are two types of signal generator, software and hardware. A software signal generator is software that capable of sending a signal down a cable so that it can be traced by a detector receiving signals on the same frequency.

A hardware signal generator is connected to some DAC (Digital Analog Converter). The number of DAC that is intends to be connected is depends on the number of bits supported by the signal generator. The DAC have the ability to convert digital data into analog data. As a result, the output will be in analog form. However, signal generator can generate digital output as well. Sometimes the output can be generated in combination of analog and digital. Each output is configured via its Setup menu. The DAC outputs may be swapped with a single toggle, or the same signal may be fed to both at once.

The signal generator has other type of source as well. It is may be connected to a sound card that can generate sound signal. The general sources are as follow:

- White noise, uniform distribution
- Gaussian (normal) distribution white noise

  (Adjustable Standard Deviation)

- Pink noise
- Band-limit noise

  (Adjustable band/gap edge frequencies and shapes)

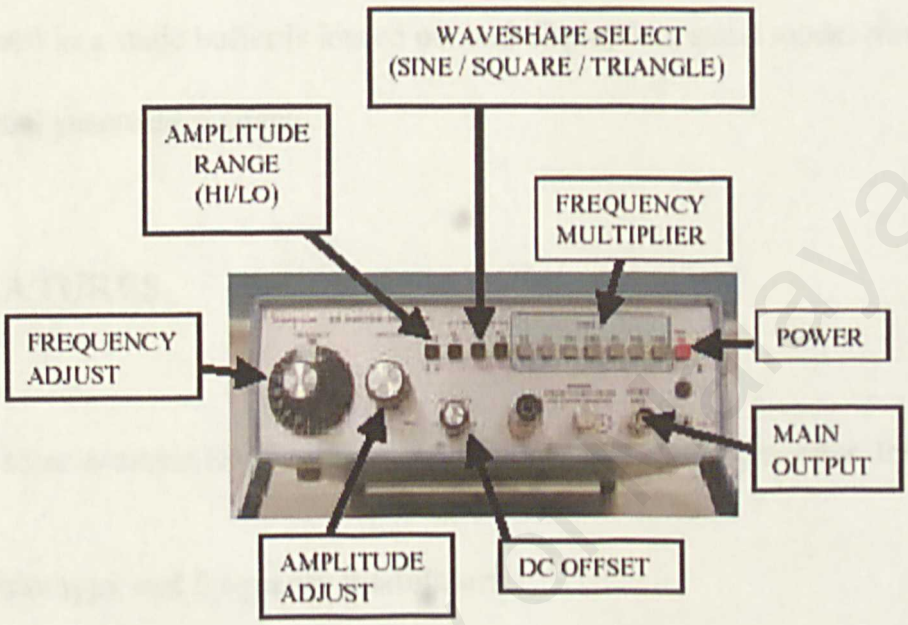


Diagram 2.4.1A: Signal generator hardware [ELEC 2010, 2002]

2.4.2 MODE

- **Dynamic mode**

Signal generator consists of dynamic mode [Mautin, 2001]. The dynamic mode permits continuous real-time signal generation. This dynamic mode not only allows extremely fine frequency resolution, but it also allows extremely long tone bursts and frequency sweeps (hours or days) and complex signal interactions. For example, two component frequencies that are set to differ by

0.0001 Hz will only be in phase once every 2.78 hours. Modulation cycles can be combined to get even longer cycles, many millions or even billions of years.

- **Static mode**

  Static mode permits static signal generation [Mautin, 2001]. The data that is stored in a static buffer is loaded out and display in a static mode. Normally the signal generated is simple.

2.4.3 FEATURES

There are some example features that conclude in general signal generator, for example:

- **Wave type and frequency modulators**

  Each DAC channel allows some different wave components to be generated, each with its own separate submenu page system. Each component page has submenu controls for wave type and frequency modulators.

- **Complete signal configurations**

  Complete signal configurations may be saved to files for automatic load on start-up, or may be saved or loaded at any time during operation.

- **Output voltage**

  Output voltage adjustable from 0 to 12 volts.

- **Power On presets**

  Power On presets, recall power on presets button

2.4.4 OPERATION

The signal generator works in synchrony with the data acquisition process. At each sample time, an output sample is given from each active stimulus source in the order DAC 0, DAC 1, DigOut before the analog input sample data is read. The stimulus samples are pre-computed and stored, in the same sequence they will be used, in one memory buffer. The size of buffer is varied depends on the number of output bits. Whenever any stimulus parameter is changed, the entire stimulus buffer is recomputed. However, the usage of this buffer changes between conventional static and the new dynamic generation modes.

By controlling the timing of each component, the waveform components can be made to occur sequentially or concurrently during a single data acquisition sweep. The signal generate by signal generator may also be grouped for output on two separate alternating sweeps or four sequential sweeps.

Generally, the signal generator generate signal through 2 process, digital frequency generation and analog processing of signal [Mautin, 2001].

- **Digital Frequency Generation**

  The digital part of the frequency generator consists of a shift register, an adder, a latch and a 1M bit EPROM. The shift register reduces the need of port pins at

the MCU from 33 to 3. It also synchronizes the data input from the MCU with the operation of the adder.

The adder output is feedback to itself via the latch. Therefore the value at the output of the latch is increased by the value in the shift register at every clock cycle. This value is also taken as an address for the EPROM. This EPROM contains a table, which allows converting the value from the latch into the amplitude of the output signal. In principle any waveform can be stored and therefore generated.

The accuracy of the generated frequency is determined at low frequencies by the precision of the oscillator and at high frequencies by the jitter, which is caused by the discrete nature of adder and table.

The frequency and amplitude modulation is based on a DDS software in the micro controller. Since the sinus is read from a 16K byte lookup table (LSB first) starting at &H01000, we can replace it by any other waveform.

- **Analog Processing of the Signal**

The final output signal is formed through DAC, filter, multiplier and an amplifier. The digital output of the EPROM is converted to a analog signal by a high speed, high precision DAC. Its current output is converted to a voltage by a operational amplifier. To remove glitches and harmonics a Butterworth filter is employed.

The MCU shifts the amplitude value into the DAC. The output of the DAC is feed into the multiplier and allows the adjustment of the amplitude. The offset is adjusted by another DAC. Its output is converted to a bipolar output by a dual OP. In the final step the signal is amplified by a strong operational amplifier.

2.5 PARALLEL PORT

What is the "parallel port"? In the computer world, a port is a set of original lines that the microprocessor, or CPU, uses to exchange data with other components [Jan Axelson, 1999]. Typical uses for ports are communicating with printer, modems, keyboards, and display, or just about any component or device except system memory. Most computer ports are digital, where each signal, or bit, is 0 or 1. a parallel port transfers multiple bits at once, while a serial port transfers a bit at time (though it may transfer in both directions at once).

The original computer's parallel port had 8 output, 5 inputs, and 4 bi-directional lines. These are enough for communicating with many types of peripherals. On many newer computers, the 8 outputs can also serve as inputs, for faster communication with scanners, drivers, and other devices that send data to the computer.

The parallel port was designed as a printer port, and many of the original names for the port's signals (PaperEnd, AutoLineFeed) reflect that use. But these days, you can find all kinds of things besides printers connected to the port. The term peripheral, or peripheral device is a catch-all category that includes printers, scanners, modems and other devices that connect to a computer.

2.5.1 PORT TYPES

As the design of computer evolved, several manufacturers introduced improved versions of the parallel port. The new port types are compatible with the original design, but add new abilities, mainly for increased speed.

2.5.1.1 ORIGINAL (SPP)

The parallel port in the original IBM PC, and any port that emulates the original port's design, is sometimes called the SPP, for standard parallel port, even though the original port had no written standard beyond the schematic diagrams and documentation for the IBM PC. Other names used are AT-type or ISA-compatible.

The port in the original PC was based on an existing Centronics printer interface. However, the PC introduced a few differences, which other systems have continued.

SPP can transfer 8 bits at once to a peripheral, using a protocol similar to that used by the original Centronics interface. The SPP doesn't have a byte-wide input port, but for PC-to-peripheral transfers, SPPs can use a Nibble mode that transfers each byte 4 bits at a time. Nibble mode is slow, but has become popular as a way to use the parallel port for input.

2.5.1.2 PS/2-TYPE (SIMPLE BI-DIRECTIONAL)

An early improvement to the parallel port was the bi-directional data port introduced on IBM's model PS/2. The bi-directional port enables a peripheral to transfer 8 bits at once to a PC. The term PS/2-type has come to refer to any parallel port that has a bi-directional data port but doesn't support the EPP or ECP modes described below. Byte mode is an 8-bit data transfer protocol that PS/2-type ports can use to transfer data from the peripheral to the PC.

2.5.1.3 EPP

The EPP (enhanced parallel port) was originally developed by chip maker Intel, PC manufacturer Zenith, and Xircom, a maker of parallel-port networking products. As on the PS/2-type port, the data lines are bi-directional. An EPP can read or write a byte of data in one cycle of the ISA expansion bus, or about 1 microsecond, including handshaking, compared to four cycles for an SPP or PS/2-type port [Peacock, 2002]. An EPP can switch directions quickly, so it's very efficient when used with disk and tape drives and other devices that transfer data in both directions. An EPP can also emulate an SPP, and some EPPs can emulate a PS/2-type port.

2.5.1.4 ECP

The ECP (extended capabilities port) was first proposed by Hewlett Packard and Microsoft. Like the EPP, the ECP is bi-directional and can transfer data at ISA-bus speeds. ECPs have buffers and support for DMA (direct memory access) transfers and data compression. ECP transfers are useful for printers, scanners and other peripherals that transfer large blocks of data. An ECP can also emulate an SPP or PS/2-type port and many ECPs can emulate an EPP as well.

2.5.1.5 MULTI-MODE PORTS

The parallel port uses a variety of the computer's resources. Every port uses a range of a address, though the number and location of addresses varies. Many ports have an assigned IRQ (interrupt request) level and ECPs may have an assigned DMA channel.

The resources assigned to a port can't conflict with those used by other system components, including other parallel ports.

2.5.2 PARALLEL PORT RESOURCE

The parallel port uses a variety of the computer's resources [Peacock, 2001]. Every port uses a range of addresses, though the number and location of addresses varies. Many ports have an assigned IRQ (interrupt request) level and ECPs may have an assigned DMA channel. The resources assigned to a port can't conflict with those used by other system components, including other parallel ports.

2.5.2.1 ADDRESSING

Each printer port consists of three port addresses; data, status and control port. These addresses are in sequential order. That is, if the data port is at address 0x0378, the corresponding status port is at 0x0379 and the control port is at 0x037a.

The following is typical.

Printer	Data Port	Status	Control
LPT1	0x03bc	0x03bd	0x03be
LPT2	0x0378	0x0379	0x037a
LPT3	0x0278	0x0279	0x027a

2.5.2.2 INTERRUPTS

Most parallel ports are capable of detecting interrupt signals from peripheral. The peripheral may use an interrupt to announce that it's ready to receive a byte, or that it has a byte to send. To use interrupts, a parallel port must have an assigned interrupt-request level (IRQ).

Conventionally, LPT1 uses IRQ7 and LPT2 uses IRQ5. But IRQ5 is used by many sound cards and because free IRQ levels can be scarce on a system, even IRQ7 may be reserved by another device. Some ports allow choosing other IRQ levels besides these two.

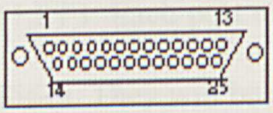
Many printer drivers and many other applications and drivers that access the parallel port don't require parallel-port interrupts. If user selects no IRQ level for a port, the port will still work in most cases, though sometimes not as efficiently and you can use the IRQ level for something else.

2.5.2.3 DMA CHANNELS

ECPs can use direct memory access (DMA) for data transfers at the parallel port. During the DMA transfers, the CPU is free to do other things, so DMA transfers can result in faster performance overall. In order to use DMA, the port must have an assigned DMA channel, in the range 0 to 3.

2.5.3 PORT HARDWARE

2.5.3.1 PC PARALLEL INTERFACE



View is looking at
Connector side of
DB-25 Male Connector.

Pin	Description	
1	$\overline{\text{Strobe}}$	PC Output
2	Data 0	PC Output
3	Data 1	PC Output
4	Data 2	PC Output
5	Data 3	PC Output
6	Data 4	PC Output
7	Data 5	PC Output
8	Data 6	PC Output
9	Data 7	PC Output
10	$\overline{\text{ACK}}$	PC Input
11	Busy	PC Input
12	Paper Empty	PC Input
13	Select	PC Input
14	$\overline{\text{Auto Feed}}$	PC Output
15	$\overline{\text{Error}}$	PC Input
16	Initialize Printer	PC Output
17	$\overline{\text{Select Input}}$	PC Output

Pin Assignments

Note: 8 Data Outputs
4 Misc Other Outputs

5 Data Inputs

Note: Pins 18-25 are
Ground

Diagram 2.5.3.1.A: Parallel port interface [Dage, 1999]

Many years ago, IBM designed the parallel port to drive printers. A standard 'D' 25 pin male connector was available on the back of the PC that connected to a printer. Their sole purpose was to interface with the facto Centronics printer. Instead of building a clean direct interface that relied on software to invert signals, they used hardware inverters in a most unusual fashion. Here an

inverter, there an inverter, everywhere an inverter. But it's the standard, everybody has one so lets use it.

DOS supports up to three parallel ports that are assigned the handles of LPT1, LPT2, and LPT3. Each port requires three consecutive IO addresses to select all the possibilities. They will be referred to as Base, Base +1, and Base + 2.

Parallel port is a misnomer. Actually there are five ports, consisting of two output ports and three input ports [NI, 2003].

At base address, eight bits are available as output on pins 2-9. They are hard wired to an eight bit input at the same address. The output is latched in the usual manner with the IO write pulse and is always active. The original IBM card used a 74LS374 tri-state device that has it output enable pin hardwired active. This means the input can only read the output so it's not usable except to check that the output is correct. The original IBM card could be reworked to control the output enable pin that would then free up the input, but this is not recommended. Resistor and capacitors suppress ringing but should not limit this ports speed. When using CMOS devices driven from this output, it's recommended that pull-up resistors be used.

At base + 1, there are five input bits from D3 to D7. They are gated on the bus with IO read. Note that bit D7 (pin 11) is inverted. Software can invert this bit if necessary and will be demonstrated later. Bit D6 (pin 10) can also be used to generate a hardware interrupt. Several conditions must be met before an

interrupt occurs. This pin can be used as a data input without concerns of inadvertently causing an interrupt. If a hardware interrupt is desired, this is the pin to use.

At base + 2 several options exist. This is a four bits output or a four bit input, or can be configured as any mixture of input and output. This is possible because the output is open collector. By sending data to this port to make an output pin high, allows that pin to be driven as an input. The open collectors are pulled high with 4.7 K resistors. Open collectors are not driven high but float high due to the pull up resistor charging any capacitance in the circuit. The speed on this port will not be as fast as on the base address particularly bit D0 that has an added external capacitor. Test any application that requires maximum output speed from this port.

Interrupts are normally open collector activated with a device pulling the line low. Any card in the ISA slot can pull an interrupt line low. This parallel port card uses a different approach. Bit D4 at base + 2 controls a tri-state device. When D4 is high, pin's 10 logic state is passed to the hardware interrupt number 7 (default) or number 5. Remember that all outputs are latched and remain active. If bit D4 is set high, and input pin 10 is high, this could disable other cards from using the selected interrupt. User software program should set bit D4 low when not controlling the interrupt.

To summarize: the parallel port is capable of eight to 12 output bits, and five to nine input bits. One of the input lines can be used as a hardware interrupt. All inputs and outputs will behave as 74LS devices.

2.5.3.2 CONNECTORS

The PC's back panel has the connector for plugging in a cable to a printer or other device with a parallel-port interface. Most parallel ports use the 25-contact D-sub connector, the shell (the enclosure that surrounds the contacts) is roughly in the shape of an upper-case D. Other names for this connector are the subminiature D, DB25, D-shell or just D connector. The IEEE 1284 standard for the parallel port calls it the IEEE 1284-A connector.

2.5.3.3 CABLES

Most printer cables have a 25-pin male D-sub connector on one end and a male 36-contact on the other. Many refer to the 36-contact connector as the Centronics connectors, because it's the same type formerly used on Centronics printer. Other names are parallel-interface connector or just printer connector. IEEE 1284 calls it the 1284-B connector.

Peripherals other than printers may use different connectors and require different cables. Some use a 25-pin D-sub like the one on the PC. A device that uses only a few of the port's signals may use a telephone connector, either a 4-wire RJ11 or an 8-wire RJ45. Newer peripherals may have the 36-contact 1284-C connector.

In any case, because the parallel-port's outputs aren't designed for transmitting over long distances, it's best to keep the cable short: 6 to 10 feet, or 33 feet for an IEEE-1284-compliant cable.

2.6 BUFFER

The simplest form of digital input is shown in diagram 2.6A. The 74HC244 buffer sits between the processor and the outside world. When the processor wants to read the status of devices on the input port, the 74HC244's output enable is asserted, and data flows through the buffer and onto the data bus.

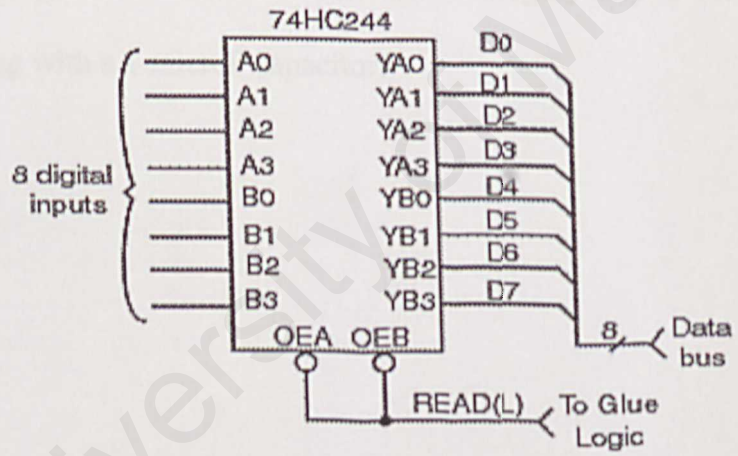


Diagram 2.6A: 74HC244 buffer [Bob Perrin, 2003]

If the system must have many digital inputs, the 74HC244 scheme may add an unacceptable level of capacitance to the microprocessor's data bus. The tri-stated output of the 74HC244 has a worst-case capacitance of 20 pF (see the 74HC244 datasheet). It doesn't take many 74HC244s on a bus to slow it down.

2.7 ANALOG TO DIGITAL CONVERTER (ADC)

The ADC0808/ADC0809 is an 8 bit ADC that also contains an 8 channel multipexer [Shauna Rae, 1999]. The ADC0808/0809 was designed for simple interface with analog inputs, especially transducers. There are a couple of limitations that follow:

- The source resistance must be below 10kohms for operation below 640kHz and below 5kohms for operation around 1.2MHhz.
- The source must remain stable while it is being sampled and should contain little noise. This means it must remain stable for up to 72 clock cycles.
- If Vcc and ground are used as reference voltages, they should be isolated by decoupling with a 1 microF capacitor.

2.8 PLATFORM

An operating system (sometimes abbreviated as "OS") is the program that, after being initially loaded into the computer by a boot program, manages all the other programs in a computer. The other programs are called applications or application programs. The application programs make use of the operating system by making requests for services through a defined application program interface (API). In addition, users can interact directly with the operating system through a user interface such as a command language or a graphical user interface (GUI).

An operating system performs these services for applications:

- In a multitasking operating system where multiple programs can be running at the same time, the operating system determines which applications should run in what order and how much time should be allowed for each application before giving another application a turn.
- It manages the sharing of internal memory among multiple applications.
- It handles input and output to and from attached hardware devices, such as hard disks, printers, and dial-up ports.
- It sends messages to each application or interactive user (or to a system operator) about the status of operation and any errors that may have occurred.
- It can offload the management of what are called batch jobs (for example, printing) so that the initiating application is freed from this work.

On computers that can provide parallel processing, an operating system can manage how to divide the program so that it runs on more than one processor at a time.

2.8.1 LINUX

Linux, a clone of the UNIX operating system that written from scratch to avoid license fees entirely, although the operation of the Linux operating system is based entirely on UNIX. It shares UNIX's command set and look-and-feel, so if anyone know either UNIX or Linux, they know the other [Tapcott, 1999].

Here are some of the important features of Linux that make it so unique [Bob Perrin, 2003]:

Full multitasking and 32-bit support

Linux is a real multitasking system that allows multiple users to run many programs on the same system at once. Linux is also a full 32-bit operating system that utilizes the special protected-mode features of Intel 80386 and later processors and their work-alike.

The X Window System

The X Window System is a very powerful graphics interface, supporting many applications. A complete version of the X Window System, known as XFree86, is available for Linux. This means Linux is moving into the GUI world in the future.

Built-in networking support

Linux uses standard TCP/IP protocols, including Network File System (NFS) and Network Information Service (NIS, formerly known as YP). By connecting the system with an Ethernet card or over a modem to another system, anyone can access the Internet.

Shared libraries and Virtual memory

Linux implements shared libraries that allowing programs use standard subroutines to find the code for these subroutines in the libraries at runtime. This saves a large amount of space on the system where each application doesn't store its own copy of these common routines.

GNU software support

Linux supports a wide range of free software written by the GNU Project, including utilities such as the GNU C and C++ compiler, gawk, groff, and so on. Many of the essential system utilities used by Linux are GNU software.

Portability

Linux is compatible with the IEEE POSIX.1 standard. Linux has been developed with software portability in mind, thus supporting many important features of other UNIX standards.

Linux is fault-tolerant

It is used to more than 31% of the world's web servers. With Apache as the primary application for these servers, they have proven to be practically immune to the recent explosion of viruses that have plagued e-mail and the Internet.

Nonproprietary source code

The Linux kernel uses no code from AT&T, nor any other proprietary source. Other organizations, such as commercial companies, the GNU project, hackers, and programmers from all over the world have developed software for Linux.

Security

Because of the available source code and the ability for users to modify, Linux is not as secure as other system if an ever-expanding group of hackers who want to get their hands dirty with others' Linux-based system.

Lower cost than most other Windows NT systems and UNIX clones systems Anyone who has the patience to access to the Internet, the only price that needs to pay for Linux is the time. Linux is freely available on the Internet.

2.8.2 WINDOWS NT

Windows NT is by far one of the most self-sufficient operating systems. It is the operating system developed by Microsoft. The most obvious part of Windows is the graphical user interface.

Describe below are some of the features of Windows NT:

Preemptive multitasking and scalability

The internals of Windows NT were centered around a microkernel-style architecture similar to UNIX. This microkernel gave Windows NT preemptive multitasking. Additionally, Windows NT made use of process threads, an idea popularized by Carnagie Mellon's MACH operating system to support symmetric multiprocessing (SMP). The internal operations of Windows NT are designed to take full advantage of SMP systems (scalability).

Flat, 32-bit Memory Model

Windows NT is a 32-bit operating system that uses 32-bit addresses to access objects. This result in many advantages such as it enables NT to address 4,194,304KB (four gigabytes) of memory.

NT Virtual DOS Machine (NTVDM)

There is no DOS in Windows NT but yet it still able to run the vast majority of DOS programs as long as they don't try to directly access the hardware or require special device drivers. It does this by creating a virtual DOS

environment called the NT Virtual DOS Machine. The DOS program runs in this emulated DOS. NT traps the DOS calls and converts them to standard Win32 API calls.

Network Operating Systems

Windows NT is both an operating system and a network operating system. With LAN Manager, OS/2 was the operating system and LAN Manager was the network operating system. This integration of the OS and the NOS has proved to be a formidable combination in Windows NT.

Reliability Through Protected Memory Model

In Windows NT's memory model all processes get their own 32-bit address space. This 4GB space is divided in half, and the application can only really use the lower 2GB of space. The upper 2GB is for interfacing with other parts of the system. Every process effectively thinks it is the only thing running. There is no way for a process to read or write outside of its own memory space, either accidentally, or intentionally. This can prevents the system crashes and it provides security for each process.

Portability

It is this portability that enables Windows NT to run not only on Intel x86 microprocessors but also on RISC chips, such as the DEC Alpha AXP, the MIPS R4400, and Motorola PowerPC. Part of the key to Windows NT's

portability is the hardware abstraction layer (HAL), which hides the difference in actual hardware from the higher-level operating system software. The HAL makes all hardware look essentially identical to the rest of Windows NT.

Personality/Compatibility

Windows NT was designed to support multiple simultaneous personalities. Its interface became the primary personality. It also supports a POSIX personality, an OS/2 personality, and a DOS/Windows personality. Additional personalities, such as a full UNIX personality can easily be added.

Security

Windows NT was created to meet the United States National Security Agency's C2 level evaluation criteria. By creating Windows NT based on a defined security model, Microsoft was able to guarantee that Windows NT would meet the most demanding corporate security needs.

Fault-Tolerance

Windows NT has many features that provide varying levels of fault-tolerance for the system. Included in NT's list of fault-tolerant features are NT's journal-based, recoverable file system (NTFS), disk mirroring and disk stripping with parity (RAID 1 and RAID 5), disk sector sparing, and support for an uninterruptible power supply (UPS).

License fees

However, Windows NT is a copyrighted piece of software that demands license fees when any part of its source code is used. Therefore, it required a sum of monetary outlay to obtain it.

2.8.3 WINDOWS 2000

Windows 2000 is a fully Web-aware operating system, with a built-in Web server, Internet Information Services 5.0. It also includes the critical application development services needed to build integrated, component-based applications that take advantage of the Internet.

The Microsoft Windows 2000 Server operating system provides the services user need to put the Internet to work for their business. From publishing basic information about company to creating a full-blown e-commerce application, starting with Windows 2000 is a great way to build the Internet into business. Using industry-standard hardware, software, and skills, along with the services and features in Windows 2000, user can readily share information and conduct transactions involving employees, customers, and business partners—anywhere in the world—through the Web.

For many companies, the ultimate goal of Internet-enabling their businesses is to create a dynamic Web-based storefront to serve customers online. With its application services, scalability, reliability, manageability, and security, Windows 2000 serves as a solid yet flexible foundation for securely integrate e-commerce sites. Using Windows 2000, developers have the ability to create virtually any type of shopping experience.

To provide maximum programming flexibility, Windows 2000 uses a component object-based programming model that lets developers use a broad array of Microsoft and third-party development tools to create applications and integrate them with existing software. The model is also programming language-neutral, so developers can use virtually any language they prefer.

2.9 PROGRAMMING LANGUAGE

2.9.1 VISUAL BASIC

The Visual refers to the method used to create the graphical user interface (GUI). Rather than writing numerous lines of code to describe the appearance and location of interface elements, you simply add pre-built objects into place on screen [Allen, 1998].

The Basic refers to the BASIC (Beginners All-Purpose Symbolic Instruction Code) language, a language used by more programmers than any other language in the history of computing.

Visual Basic contains several hundred statements, functions, and keywords, many of which relate directly to the Windows GUI. Beginners can create useful applications by learning just a few of the keywords, yet the power of the language allows professionals to accomplish anything that can be accomplished using any other Windows programming language.

The Visual Basic programming language is not unique to Visual Basic. The Visual Basic programming system, Applications Edition included in Microsoft Excel,

Microsoft Access, and many other Windows applications uses the same language. The Visual Basic Scripting Edition (VBScript) is a widely used scripting language and a subset of the Visual Basic language. Listed below are the features of Visual Basic:

- Data access features allow user to create databases, front-end applications, and scalable server-side components for most popular database formats, including Microsoft SQL Server and other enterprise-level databases.

- ActiveX technologies allow you to use the functionality provided by other applications, such as Microsoft Word word processor, Microsoft Excel spreadsheet, and other Windows applications. User can even automate applications and objects created using the Professional or Enterprise editions of Visual Basic.

- Internet capabilities make it easy to provide access to documents and applications across the Internet or intranet from within user application, or to create Internet server applications.

- User's finished application is a true .exe file that uses a Visual Basic Virtual Machine that they can freely distribute.

2.9.2 JAVA

A simple, object-oriented, network-savvy, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, dynamic language. Below are the features of Java language.

- **Simple**

  Java omits many rarely used, poorly understood, confusing features of C++ that in our experience bring more grief than benefit. These omitted features primarily consist of operator overloading (although the Java language does have method overloading), multiple inheritance, and extensive automatic coercions.

  Automatic garbage collection, simplifies the task of Java programming but making the system somewhat more complicated. By virtue of having automatic garbage collection (periodic freeing of memory not being referenced) the Java language not only makes the programming task easier, it also dramatically cuts down on bugs.

  Another aspect of being simple is being small. A small size is important for use in embedded systems and so Java can be easily downloaded over the net.

- **Object-Oriented**

  Simply stated, object-oriented design is a technique that focuses design on the data or objects. Object-oriented design is also the mechanism for defining how

modules "plug and play." The object-oriented facilities of Java are essentially those of C++, with extensions from Objective C for more dynamic method resolution.

- **Network-Savvy**

  Java has an extensive library of routines for coping easily with TCP/IP protocols like HTTP and FTP. This makes creating network connections much easier than in C or C++. Java applications can open and access objects across the net via URLs with the same ease that programmers are used to when accessing a local file system.

- **Robust**

  Java is intended for writing programs that must be reliable in a variety of ways. Java puts a lot of emphasis on early checking for possible problems, later dynamic (runtime) checking, and eliminating situations that are error prone.

  Java has a pointer model that eliminates the possibility of overwriting memory and corrupting data. Instead of pointer arithmetic, Java has true arrays. This allows subscript checking to be performed.

  Java programmers can be relatively fearless about dealing with memory because they don't have to worry about it getting corrupted. Because there are no pointers in Java, programs can't accidentally overwrite the end of a memory buffer.

- **Secure**

  Java is intended for use in networked/distributed environments. Toward that end, a lot of emphasis has been placed on security. Java enables the construction of virus-free, tamper-free systems. The authentication techniques are based on public-key encryption.

  There is a strong interplay between "robust" and "secure." For example, the changes to the semantics of pointers make it impossible for applications to forge access to data structures or to access private data in objects that they do not have access to. This closes the door on most activities of viruses.

- **Architecture Neutral**

  Java was designed to support applications on networks. In general, networks are composed of a variety of systems with a variety of CPU and operating system architectures. To enable a Java application to execute anywhere on the network, the compiler generates an architecture-neutral object file format--the compiled code is executable on many processors, given the presence of the Java runtime system.

- **High Performance**

  While the performance of interpreted bytecodes is usually more than adequate, there are situations where higher performance is required. The bytecodes can be translated on the fly (at runtime) into machine code for the particular CPU the application is running on. For those accustomed to the normal design of a

compiler and dynamic loader, this is somewhat like putting the final machine code generator in the dynamic loader.

The bytecode format was designed with generating machine codes in mind, so the actual process of generating machine code is generally simple. Efficient code is produced: the compiler does automatic register allocation and some optimization when it produces the bytecodes.

- **Multithreaded**

Multithreading is a way of building applications with multiple threads. Unfortunately, writing programs that deal with many things happening at once can be much more difficult than writing in the conventional single-threaded C and C++ style.

Java has a sophisticated set of synchronization primitives that are based on the widely used monitor and condition variable paradigm introduced by C.A.R.Hoare. By integrating these concepts into the language (rather than only in classes) they become much easier to use and are more robust.

Other benefits of multithreading are better interactive responsiveness and real-time behavior.

2.9.3 C++

What is it?

Nearly all world-class software, from the leading Web browsers to mission-critical corporate applications, is built using the Microsoft Visual C++ (VC++) development system," gushes Microsoft's introduction to Visual C++ 6.0 Professional Edition.

However, this will certainly be news to the vast number of corporations whose core systems are written in Cobol. And while, post-Cobol, C and C++ may be the languages of choice, and VC++ the obvious choice for Windows developments, it is far from the only C++ development tool.

Where did it originate?

C++ was initially designed and implemented by Bjarne Stroustrup at AT&T Labs (then AT&T Bell Labs). It evolved from C, with the addition of object-oriented capabilities from the Simula programming language.

The first commercial release was in 1985. The language gained widespread use in industry and academia during the 1980s, and in 1990 the major computer and software tools suppliers, Microsoft among them, started to provide C++ to their users as a major implementation tool.

C++ is used by more than 1.5 million programmers worldwide. Apart from Microsoft and AT&T, companies that have contributed to the C++ standard include Ericsson, Borland, HP, IBM, Silicon Graphics and Sun.

What's it for?

C++ is a general-purpose programming language with a bias towards systems programming. It supports low-level programming in traditional styles, data abstraction, object-oriented programming, and generic programming.

Visual C++ is a C++ development environment for Windows and the Web, including scripting, compiling and debugging tools and component libraries. User can develop applications that make use of OLE (Object Linking and Embedding), ODBC (Open Database Connectivity) and the Microsoft Foundation Class (MFC) library. VC++ can be used to build ActiveX controls, and create multimedia-based, interactive, Dynamic HTML (DHTML) pages.

3.0 SUMMARY

This chapter discusses the literature reviews regarding the system architecture, system platform, database system, programming and also development tools that are done. After doing those research, I will choose the most suitable one in developing my proposed system. I will further explain about the software chosen and also the system requirement in Chapter 3.

CHAPTER 3: SYSTEM ANALYSIS

As progress from the literature research and review stage, the next phase is to perform a thorough and detailed analysis of the system development. Emphasis will be on the functional and non-functional requirements of the system development needs. Thus system analysis is a stage of gathering, analyzing and finalizing or determine the problem scope and the objective of goals which the system will have to achieve.

3.1 METHODOLOGY

3.1.1 INTRODUCTION

To ensure the success, a good project planning must be done. With a good project planning system can meet the requirement and the expectation of the users. Project planning also must be constantly monitored throughout the project and periodically updated based on the most recent information.

A methodology may be defined as a collection of procedures, techniques, tools and documentation aids that can help to speed up and simplify the development process. Methodology also will help to plan, manage, control and evaluate information system project. The objectives of the methodology include the following:

1. record accurately the requirements for an information system
2. provide a systematic method of development so that progress can be monitored
3. provide an appropriate time limit and acceptable budget
4. produce a system that is well documented and easy to maintain

5. provide an indication of needed changes as early as possible in the development system
6. provide a system that is user-friendly

3.1.2 SYSTEM DEVELOPMENT LIFE CYCLE (SDLC)

Every system must go through SDLC to meet the success of the project. Every system must go through the same phase in their lifetime. The phases are:

1. feasibility study
2. analysis and requirement specification
3. design
4. implementation
5. maintenance

The process in the life cycle always consist of a set of steps: methods, tools and procedures. The model is chosen based on the nature of the project and applications, the methods and tools to be used and the controls deliverables that are required.

Using prototyping model will do the suitable model for this project.

3.1.3 DEVELOPMENT APPROACH

Software engineering process consists of a set of steps that encompass methods, tools and procedures. These steps are often referred as software engineering model or software life cycle model.

Rapid prototyping is a process that enables the developer to create a model of the software to be built. It is a working model that is functionally equivalent to a subset of the target software. The subset usually consists of data input screens (or forms), user interface (menus, dialog) and reports.

System developer must first build a rapid prototype and then lets the user interact and experiment with it. If the rapid prototype does what user wants and the user will happy with it. This will draw up a specification with the assurance that the product meets the user real needs. Then the software process can be continuing on the design and implementation stages.

Two forms that are include in rapid prototyping:

1. a model that depicts human computer interaction in a form that enables the user to understand how such interaction will occur
2. a working prototype (functional prototype), which implements some of the functions, that is require of the desired software.

3.1.4 REASONS FOR THE SELECTED APPROACH

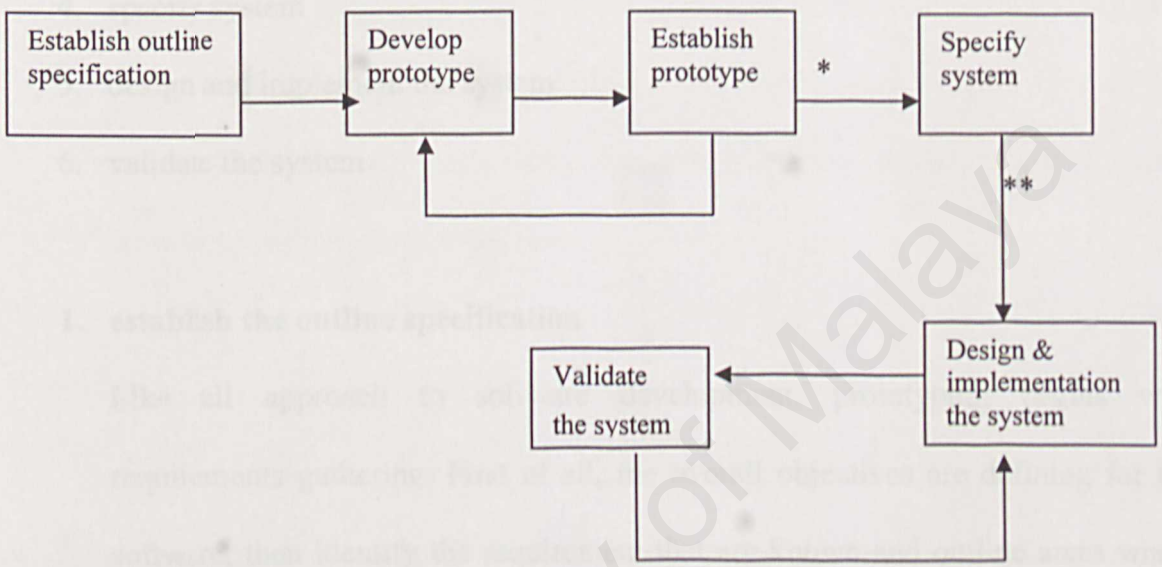
The methodology of prototyping has been chosen for this system development process. The main purpose of the prototyping method is the development and maintenance of complete information system, by means for the application of structured techniques to the enterprise as a whole. This method also emphasizes on planning, modeling and system design of the project.

Prototyping supports the entire software development process within the scope of a phase concept with the following main phases: strategic information planning, user-level system analysis, technical system design and implementation. Prototyping requires a development of application in which all the results derived from the various methods are counterchecked and consolidated so the development of application is comprised of assessed and consistent information.

This approach also has obvious advantages, particularly for a project domain, activities and entities that will serves as a basis of writing the specification for a production quality system. Thus with this approach it will capture user's requirements accurately and completely is vitally important to successful information system development. Prototype also can help users to express their requirements more precisely.

3.1.5 DESCRIPTION OF METHODOLOGY DEVELOPMENT

The sequence of events for the throw-Away prototyping model is illustrated in diagram 3.1.5A.



* at this point the prototype is thrown away

** the development of the actual system begins

Diagram 3.1.5A: Prototyping model

The sequence of events consists:

1. establish the outline specification
2. develop prototype
3. evaluate prototype
4. specify system
5. design and implement the system
6. validate the system

1. establish the outline specification

Like all approach to software development, prototyping begins with requirements gathering. First of all, the overall objectives are defining for the software, then identify the requirement that are known and outline areas where further definition is mandatory. In this activity is to help to understand the content and the complexity of the project. The mains objective here is to identify the resource requirements for each activity in the project development and provide an estimate of resources required. Project schedule must be prepared, this is to make sure that the project will complete in the time needed. Creating the preliminary budget is important to determine the planned expenses and revenues associated with the project. The baseline project plan is used to reflect the best estimate of the project tasks and the resource requirements. This will help to guide the next phase of the project development.

2. **develop prototype**

a quick design then occurs. The quick design focuses on representation of those aspects of the software that will be visible to the user (for example inputs, user interface and output). The quick design leads to the rapid construction of the prototype. The important thing that to do are just concentrate about the software that will use in this project and to know the enhancement of the software. The reason why it need a quick design because it is easy to understand, delegate, code test, document and maintain.

3. **evaluate prototype**

The prototype is evaluated by the user and is used to refine requirements for the software to be developed. Besides functions and descriptions of the prototype will be evaluate to see whether it can meet the aims and objective of the system. Some changes may be needed, this is to ensure that the system will meet all the requirements.

4. **specify system**

a process of iteration occurs as the prototype is turned to satisfy the needs of the user while at the same time enabling the developer to better understand what needs to be done. In this phase all the strength and weaknesses of the system will be discuss. Some of the functions will be added in the system. The objective are:

1. To help to capture user requirements more accurately.
2. It helps to clarify and define an ill-structured system more clearly.
3. Helps to explore the feasibility of the application.

The prototype is used as a tool for requirement analysis. After evaluation the prototype is thrown away and the actual production of the operational system begins.

5. Design and Implement the System

When the prototype enter the design phase for the second time this means the prototype need some editing from the previous work. Thus, this will improve the quality of the design. System implementation includes coding, testing and documenting the system as well as training the end users and system administrators. Design is coded with the suitable computer language such as C++, Visual Basic or Java. For the ease of the maintenance, the program module must be properly structured. After that the system must be tested. The purpose of testing is to uncover the software error. With the proper testing this will minimize the number of error. In this phase documentation is important documentation must also be provided in the form of user and operator manuals as well as system documentation.

6. **Validate the system**

At last the complete system will be tested and no more prototyping is involve. In this phase, keeping the system working in dynamically changing environment is needed. The systems also need to be modified because of the changes in the environment such as changing of the hardware and operating system.

3.1.6 THE ADVANTAGES OF THE PROTOTYPING

The benefits of developing a prototype in the software process are:

1. Misunderstanding between software developers and users may be identified as the system functions for the logic analyzer and signal generator are demonstrated.
2. Missing user services may be detected to satisfy the user need such as to add some logic analyzer and signal generator applications.
3. Any difficulty or confusing of user services may be identified and refined every time user need some changes.
4. Easy to find incomplete or inconsistent requirement as the prototype is developed, this will allow to a good modification and this would be easier to do some changes in the logic analyzer and signal generator project.
5. A working system is available quickly to demonstrate the feasibility and the usefulness of operation to management.

3.2 ANALYSIS PROCEDURES

An analysis procedure is important in order to identify the various requirement and objectives that need to be met by the user. The outcome of the analysis will serve as guideline for the developer during the design on what and how the system has to function in order to fulfill the user require. Listed below are the procedures for system analysis:

i) Problem Identification

ii) Evaluation And Synthesis

iii) Modeling

iv) System Requirements And Specification

3.2.1 PROBLEM IDENTIFICATION

Essentially the first phase in the system analysis procedure, the problem identification stage is being carried out in order to recognize and identify the objective of the system and the goals that the system need to meet in order to satisfy the user needs. In logic analyzer module the problem identification is that the system has to display input by user from parallel port pins. User has to know how does the data can be display in square waveform. The signal generator, on the other hand, meet difficulty of how to create interface that let user generates the data and transfer it to parallel port pins. Difficulties of how to write and read data to and from a parallel port are another problems occur during developing this system. The main goal that needs to be met by

the system is to be able to function according to clock cycle as well as to provide the user a reliable, user friendly and efficient functionality.

3.2.2 EVALUATION AND SYNTHESIS

In this stage, analyzing of problems has to be done by dividing the problem scope into smaller parts so that the problem can be tackle one by one. By doing this, user can have a better view and understanding on the problem scope and eventually easier to tackle and solve.

The following are problems that needed to be considered for system requirement:

i) Consideration of programming language that will suite the development of the system.

ii) Consideration of how to create certain functions of logic analyzer and signal generator.

iii) Consideration on how to read data from parallel port.

iv) Consideration of how to write data to parallel port.

v) Consideration of locating data in buffer.

vi) Consideration of how to processing data according to clock cycle.

vii) Consideration of how to display data in interface.

3.2.3 MODELING

By using a graphical representation on the system design, user can better understand how the system actually works. There are many graphical models that can be use for design purpose. The graphical model include:

- Top level architecture

  Roughly display the architecture of the system. This can give a user the basic concept of the system architecture.

- Block diagram

  Show the internal connection of electronic circuit. Normally this require extra know of the functionality of electronic hardware. The wrong connection of electronic circuit will cause the failure of system.

- Flow chart

  Divide the system into smaller scope and is shown in a hierarchy diagram. This can give a brief concept to designer of what have to do during the system development.

- Pre-representation of user interface

  This is important especially for the system that involves creating a software that include certain functions.

3.2.4 SYSTEM AND USER REQUIREMENT

System requirement of the project needs to be identified to serve as a guideline during the development process. User can identify system requirement as of:

i) What tasks does the system needs to perform

ii) What are the objectives or goals does the system needs to achieve

iii) What resources does the system need to fully function

There a 2 types of requirements which are :

i) Functional requirements

ii) Non-Functional requirements

Functional Requirement

A functional requirement refers to the interaction between the system and the working environment the system resides in. It also describes how a system should behave and perform their task given a set of input or stimuli. Stated below are the components in functional requirement analysis:

A) Logic analyzer

- Clear Buffer: clears all sampled data that is in buffer
- Timing: let user selects timing
- Duration: show the period between 2 timing
- Ruler: a dotted line that functions as ruler
- Display format: display data in flatten, bus or binary format
- Close: close the interface
- User authentication: user login using his own password and id

B) Signal generator

- Pins input: simulate the 8 data pins that let the user to "on" or "off" the parallel port pins
- Input: let user to input data by typing H_L_ where H indicates high, L indicate low and _ indicate seconds
- Delete : delete data
- Close: close interface
- User authentication: user login using his own password and id

Non-Functional Requirements

Non-Functional requirement can be described as constraint where the system must operate to a certain degree of standard of operation. These constraints usually narrow the selection of development tools such as programming language, platform and others. Listed below are the non-functional requirements of the system:

i) Reliability

Reliability refers the extent of the system performance which are expected or desired by the user which required criteria such as precision and accuracy.

ii) User Friendliness

Designing of A Graphical User Interface will eventually enhance user friendliness as well as easier for the user to understand and use the system. A good consideration of User Interface design is that the interface must fulfill the following criteria:

a) Consistent or standard – which mean every interface of different module shall look similar in order not to confuse the User.

b) High Degree Of Understandability – Enable easier use of the system.

iii) integrated graphic and multimedia environment

This enable the system user to view the waveform input by the user.

iv) Serviceability

The entire of the system data and the application should available at all the time.

v) Security

The system should be equipped with sufficient security. Each access by the user should be authenticated and validated by the system. The system should not show any potential of leakage of information. The password should be encrypted.

3.2.5 RUN TIME REQUIREMENTS

Run Time Requirement refers to the specification that the system needs to function. It can be divided into 2 main categories which are:

i) Hardware Requirement

ii) Software Requirement

Hardware Requirements in Developing the System

Hardware requirement refer to the hardware support needed in order to run the system smoothly.

The minimum computer requirements are:

a) logic analyzer

1. A personal computer

   One compatible computer is needed to develop the logic analyzer application.

2. Parallel port (at least 200 kHz)
3. Transformer 0-5 volt
4. One ADC0809 (Analog to Digital Converter)
5. Two 74LS244 buffers

Table 3.2.5A: Computer's hardware requirement

Hardware requirement	Capacity/type
Processor	At least 2333 MHz
RAM	At least 32MB
Hard drive	At lest 400MB
Interface port	Parallel port
Other standard computer peripherals	

b) signal generator

1. A personal computer

   One compatible computer is needed to develop the signal generator application.

2. Logic analyzer

3. Parallel port (at least 200 kHz)

Table 3.2.5B: Computer's hardware requirement

Hardware requirement	Capacity/type
Processor	At least 2333 MHz
RAM	At least 32MB
Hard drive	At lest 400MB
Interface port	Parallel port
Other standard computer peripherals	

Software Requirements in Developing the System

Software Requirement refers to the application or software tools needed in order to develop and run the system at the server side.

- Windows 2000 server operating system and server platform
- Microsoft Visual C++ 6.0 to write coding of user interface and input/output operation
- Microsoft Office 2000 to write the documentations

3.2.5.1 OPERATING SYSTEM

Windows 2000

After the literature review phase, it was proposed that this system would be developed using Windows as the platform. Windows 2000 includes fully integrated Internet-enabled application development technologies.

Windows 2000 supports greatly improved reliability, reduced number of required reboots, increased networking performance, increased security, and greatly improved management services.

Besides that, Windows 2000 was chosen due to the following features :

* **Reliability**

  Windows 2000 Web and Application services easier to restart services and it can even automatically restart itself if a bad Web application does cause a crash. In addition, IIS 5.0 supports an improved application protection model to help make sure that bad applications can't crash the Web server.

* **Scalability**

  With Windows 2000, user can operate everything from a single-server Web site to an enormous site running on a room full of Web servers with higher performance and workload requirements by increasing computer's scalability. User also can scale up by adding processors and memory or by purchasing larger servers. User can use

clustering services to connect multiple servers together and network load balancing to distribute the work across the servers.

* **Manageability**

Windows 2000 offers greatly simplified local and remote management, which increases an administrator's flexibility and helps reduce the amount of time spent managing the system. In addition, it is easier to run multiple sites from a single server, and user can delegate administration tasks so different people can manage different Web sites on the entire server.

* **Security**

Windows 2000 security services provide an integrated, comprehensive, and interoperable security solution for protecting your most sensitive applications and data. With Windows 2000, application developers can centralized security services for object naming and location, user authentication, single sign-on, and centralized configuration and policy management.

* **Software and data integration**

Windows 2000 Server supports a full range of industry standards to allow user to build applications that integrate with databases, mail servers, and other existing systems. Windows 2000 is particularly important for integrating information from different sources. Data format for structured document interchange on the Web gives developers a standard way to use information regardless of how it was created.

Developers also have a number of options for working with applications running on other operating systems. Windows 2000 also includes technologies that give user easy access to virtually any type of data source.

* **Hardware Support**

Windows 2000 lets you take advantage of the latest hardware to make user site even faster and more secure. Commerce accelerators are hardware cards that user plugs into their servers to speed up the secure socket layer (SSL) processing. The accelerator offloads the specialized encryption processing required for SSL processing and frees up the processor to execute user business logic.

3.2.5.2 PROGRAMMING LANGUAGE

C++ fully support object-oriented programming, including the 4 pillars of object-oriented development: encapsulation, data binding, inheritance and polymorphism. These pillars emphasize the creation of reusable software component by "crafting valuable classes".

C++ improves on many of C's features and provides object-oriented-programming (OOP) capabilities that hold great promise for increasing software productivity, quality and reusability.

3.2.5.3 SOFTWARE TOOL

Microsoft Visual C++ 6.0

What makes it special?

Microsoft says that by using VC++6, developers will spend less time building applications and less time coding, compiling and debugging. They also benefit from greater component reuse.

While competing IDEs (Integrated Drive Electronics), such as Borland's C++Builder and IBM's Visual Age for C++, may offer the same or better capabilities, the ubiquity of Microsoft technologies makes VC++ a good, safe, bread-and-butter skill [Nick, 2000]

3.3 SUMMARY

This chapter describes the planning process for the development of the software based logic analyzer and signal generator, based upon the project methodology—Rapid Prototyping model. It also explain in detail the requirements of the system, ranging from functional, non-functional to hardware and software requirements. Besides that, development tools are also has been elaborated. The next chapter will continue to give a clearer picture of the system by showing the system structure and design, flow chart and the user interface.

CHAPTER 4: SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

A) LOGIC ANALYZER

The logic analyzer is built by connecting a switch to a personal computer through parallel port. The diagram 4.1A shows the top level design of logic analyzer. First, user input data by connecting the transformer to data pins on parallel port. User can only activate pin D0 through D7. The output will show on a interface on the computer screen.

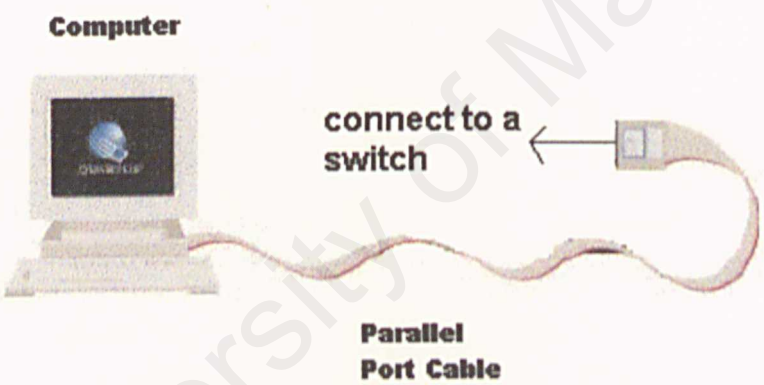


Diagram 4.1A: Logic analyzer top level diagram

B) SIGNAL GENERATOR

The signal generator, on the other hand, is built by connecting a computer to a logic analyzer through parallel port. Diagram 4.1B shows the top level design of signal generator. From the computer, user can input data by pressing the button on signal generator interface. The interface have the simulation feature of a hardware signal generator. After input data on the interface, the signal will

transfer to parallel port. The parallel port will connect to a logic analyzer which will displays the output.

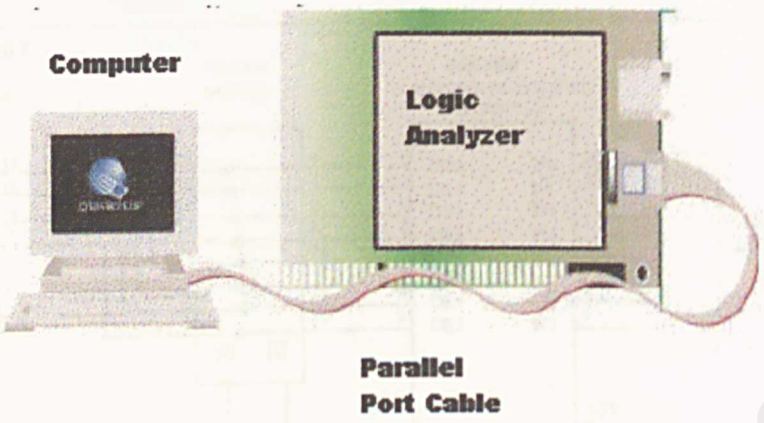


Diagram 4.1B: Signal generator top level diagram

4.2 BLOCK DIAGRAM DESIGN

A) LOGIC ANALYZER

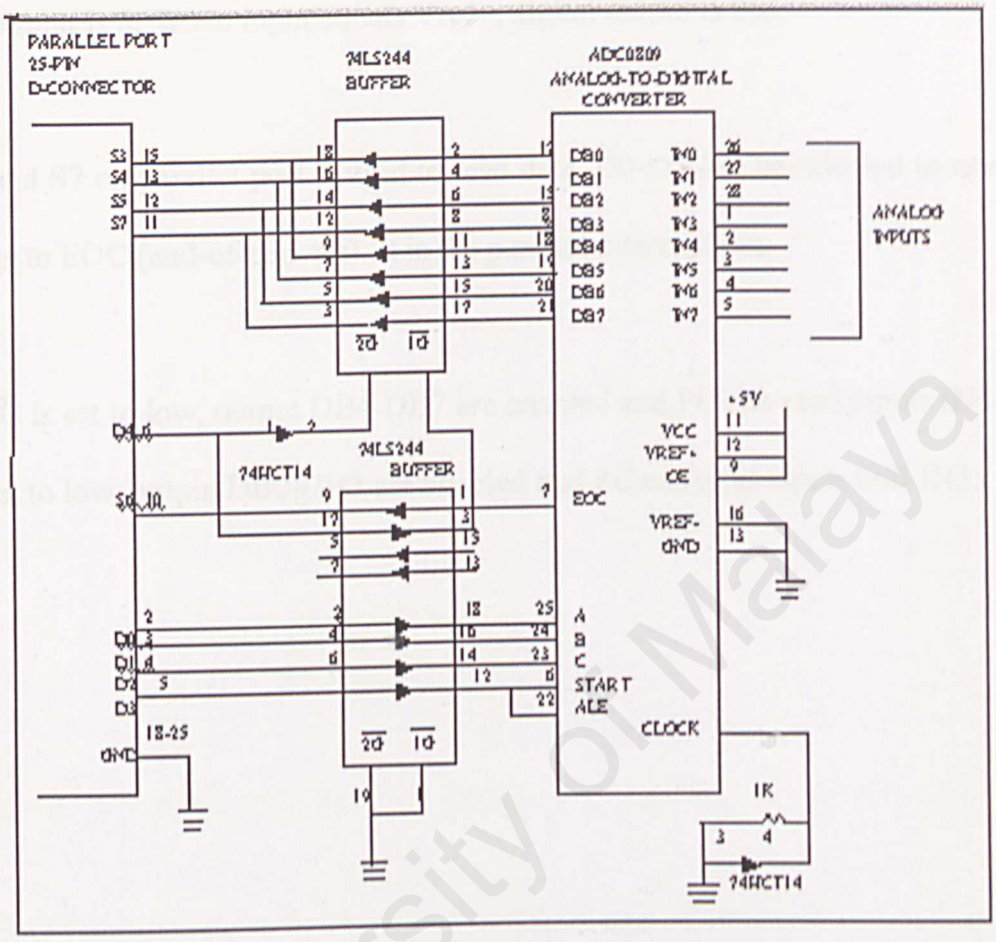


Diagram 4.2A : ADC0809 analog-to-digital converter [Jan Axelson, 1999]

ADC0809 can convert the voltage to digital value that a computer can store, display and perform calculations on. ADC0809 has 8 analog inputs (IN0-IN7), which may range from 0 volt to +5 volts. Start and ALE will start the conversion. In this system design, a clock signal is needed to control conversion of input from analog to digital.

The 74HCT14 is a Schmitt-trigger inverter. It offers a simple way to create the clock. The frequency can range from 10 kilohertz to 1280 kilohertz. Conversion time for ADC is 100 microseconds with a 640-kilohertz clock.

Vref- and Vref+ are references for analog inputs. When an analog input equals Vref-, digital output is 0. When input equals Vref+, digital output is 255.

S3-S5 and S7 on parallel port is used to read data. D0-D2 can be selected to read. S6 is connects to EOC (end-of-convert). It is the parallel interrupt pin.

If the $\overline{2G}$ is set to low, output DB4-DB7 are enabled and PC can read inputs IN4-IN7. If 1G is set to low, output DB0-DB3 are enabled and PC can read inputs IN0-IN3.

B. SIGNAL GENERATOR

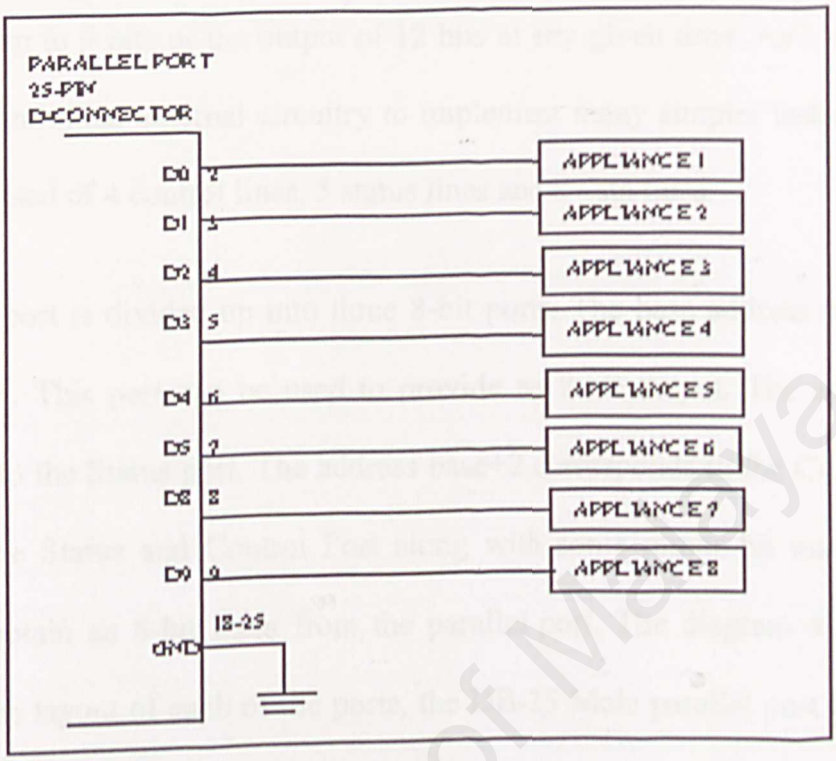


Diagram 4.2B: D0-D7 port is connected to appliances to test the outputs

Signal generator need only 1 74LS244 buffer. First, the D0 until D7 in the parallel port is connected to pin 2, 4, 6, 8, 11, 13, 16 and 17. Then another 8 pins of 74LS244 buffer are connected to logic analyzer pins. The 74LS244 buffer is used to store data temporarily from parallel port before transfer the data to logic analyzer. The block diagram 4.2B shows the hardware connection of a signal generator.

4.3 PARALLEL PORT INTERFACE

The parallel port is commonly used for interfacing project. The parallel port will allows the input of up to 9 bits or the output of 12 bits at any given time. As a result, parallel port requires minimal external circuitry to implement many simpler tasks. The parallel port is composed of 4 control lines, 5 status lines and 8 data lines.

The parallel port is divided up into three 8-bit ports. The base address corresponds to the Data port. This port can be used to provide an 8-bit output. The address base+1 corresponds to the Status port. The address base+2 corresponds to the Control Port. By combining the Status and Control Port along with some minor bit manipulation it's possible to obtain an 8-bit input from the parallel port. The diagram 4.3A, 4.3B and 4.3C show the layout of each of the ports, the DB-25 Male parallel port connector and the pin description of the DB-25 connector.

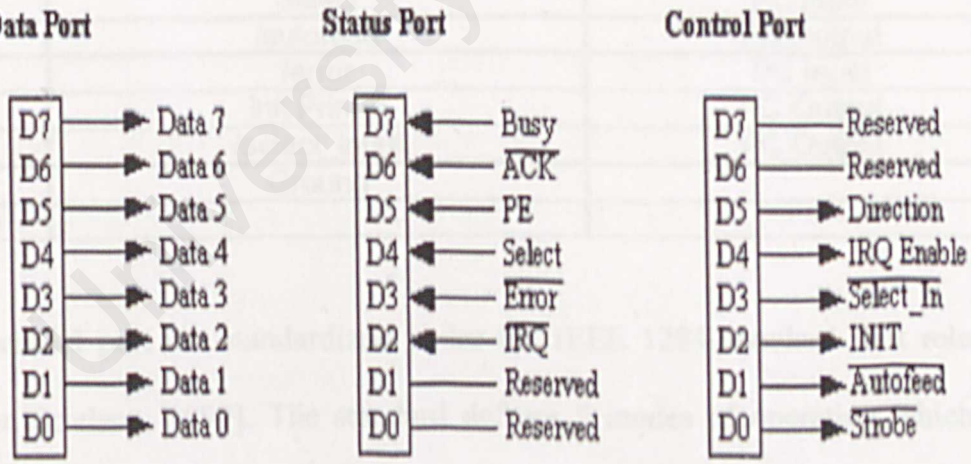


Diagram 4.3A: Parallel port types [Kyle C., 1999]

Diagram 4.3B: DB-25 male parallel port connector [Kyle C., 1999]

Table 4.3C: Parallel port pin description [Mohammed Elzubeir, 2000]

Pin	Description	Notes
1	/strobe	PC Output (OC)
2	Data_0	PC Output
3	Data_1	PC Output
4	Data_2	PC Output
5	Data_3	PC Output
6	Data_4	PC Output
7	Data_5	PC Output
8	Data_6	PC Output
9	Data_7	PC Output
10	/ack	PC Input
11	Busy	PC Input
12	Paper Empty	PC Input
13	Select	PC Input
14	/autofeed	PC Output
15	/error	PC Input
16	Init Printer	PC Output
17	/select_input	PC Output
18	Ground	
-25		

Newer parallel port are standardized under the IEEE 1284 standard first released in 1994 [Jan Axelson, 1999]. The standard defines 5 modes of operation which are as follow:

- Compatibility mode
- Nibble mode
- Byte mode

- EPP mode (Enhanced Parallel Port)
- ECP mode (Extended Capabilities Mode)

The aim was to design new drivers and devices which were compatible with each other and also backwards compatible with the Standard Parallel Port (SPP). Compatibility, Nibble and Byte modes use just the standard hardware available on the original Parallel port card while EPP and ECP modes require additional hardware which can run at faster speeds, while still being downwards compatible with the Standard Parallel Port.

4.4 OPERATION

4.4.1 READ ROUTINE

On the original parallel port [Jan Axelson, 1999],

- data port was designed as an output-only port
- the status port has 5 input
- on some ports the control port's 4 bits may be used as inputs

In the original PC's parallel port, a 74LS374 octal flip-flop drives the data outputs (D0-D7). If there were a way to disable the data port's outputs, external signals can be connected to data pins. Then the signals can be read at data port's input buffer. Some 74LS374 even has an output-enable (OE) pin. When OE is low, the outputs are enabled. When OE is high, outputs are tri-stated. On the original parallel port, OE is wired directly to ground, so the outputs are permanently enabled. Thus data ports have to be configured to bi-directional state before reading.

To read a parallel port, first port register is specified. This can be done by passing a base address to the port connector. The read routine will calculates the address to find out which port is read. Then the port connector will returns a value. Finally CPU is instructed to read the data into the requested location.

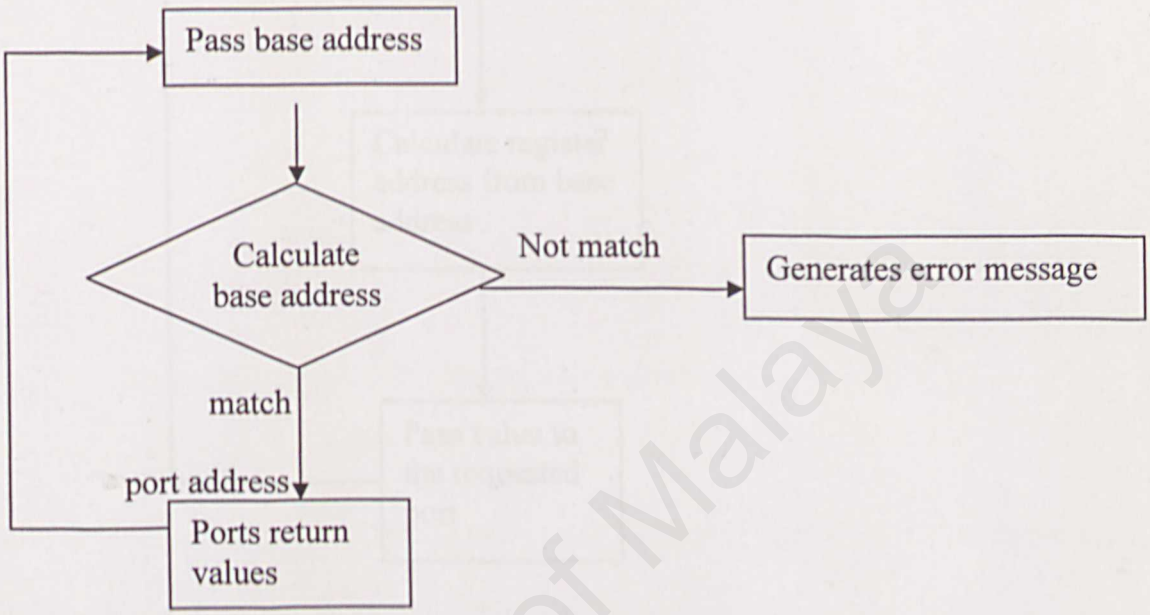


Diagram 4.4.1A: Read routine

4.4.2 WRITE ROUTINE

Data ports allow output. As a result, configuration data ports to output is not necessary. For the port-write subroutines, the base address of a port is passed to parallel port. The routines automatically calculate the register address from the base address and invert the appropriate bits. Finally, the value will be passed to the requested port [Jan Axelson, 1999].

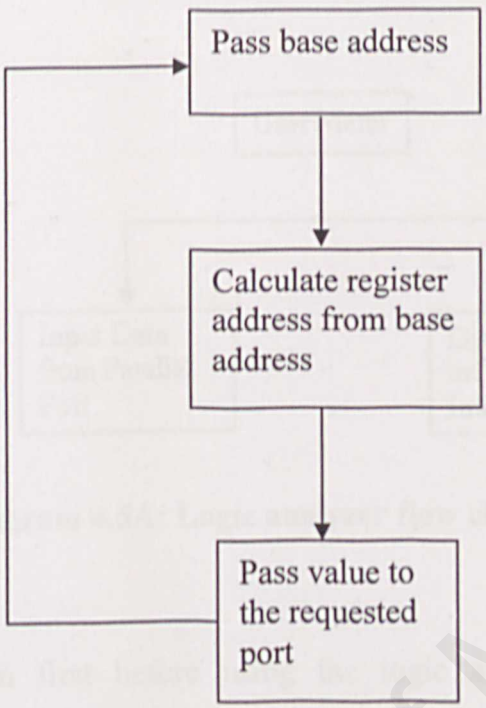


Diagram 4.4.2A: Write routine

4.5 FLOW CHART

A) LOGIC ANALYZER

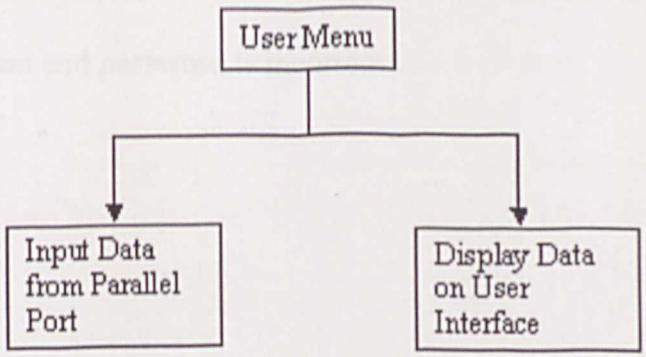


Diagram 4.5A: Logic analyzer flow chart

User has to login first before using the logic analyzer software. If their user identification and password is correct, they will be allow to use the logic analyzer menus which are input data from parallel port and display data on user interface. If their user identification and password is wrong, they have to re-login again.

B) SIGNAL GENERATOR

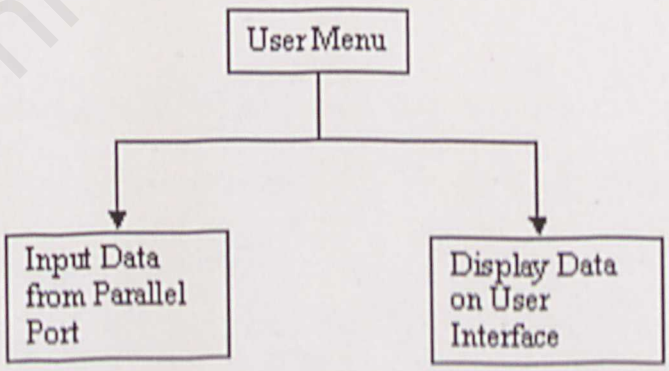


Diagram 4.5B: Signal generator flow chart

Similar to logic analyzer system, user who use signal generator software has to login themselves before using user menu. When their user identification and password is correct, they will be allowed to use the software. The user menus consist of input data from parallel port and display data on user interface. User has to re-login again when the user identification and password is incorrect.

4.6 SYSTEM FUNCTIONALITY DESIGN

A) LOGIC ANALYZER

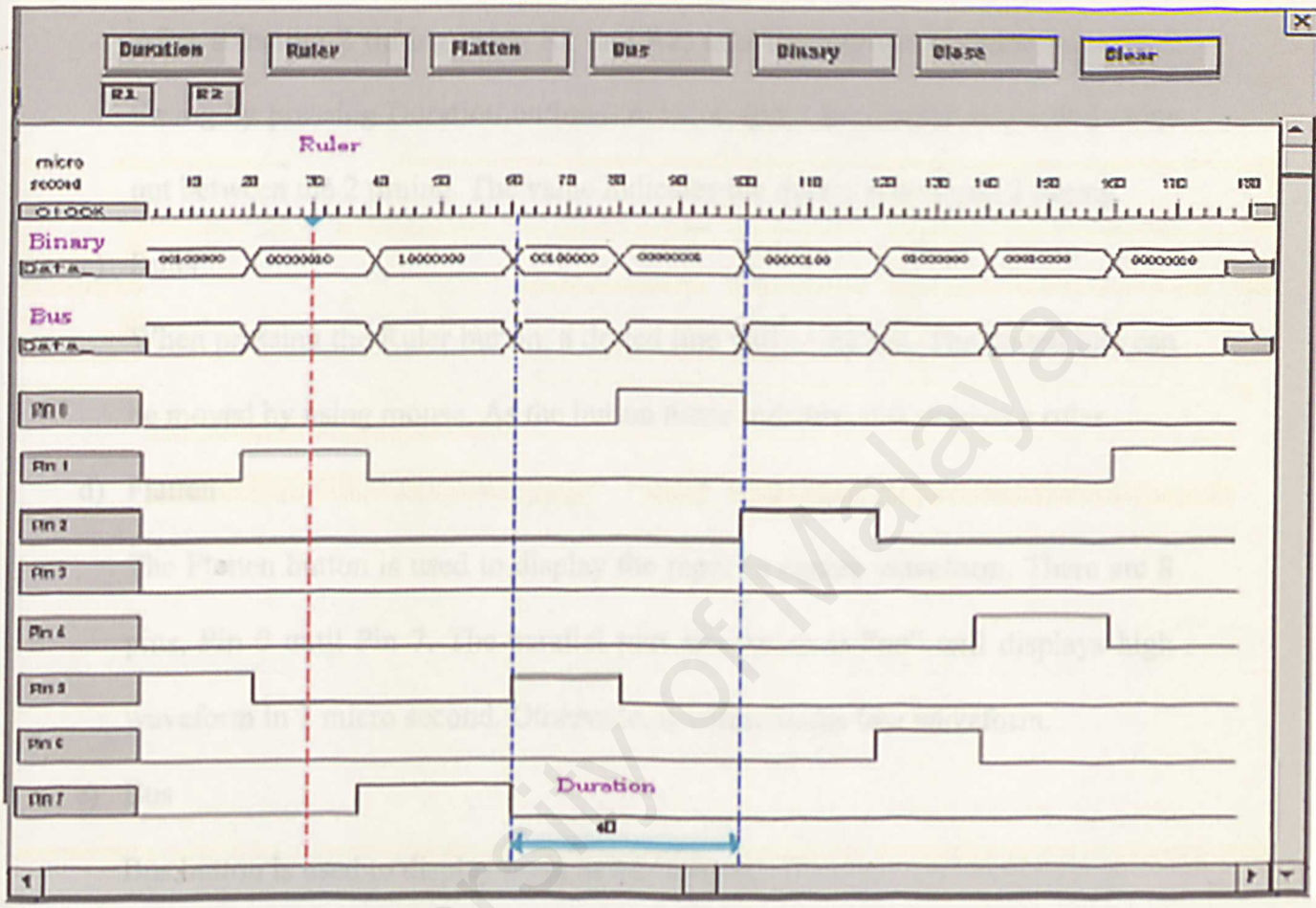


Diagram 4.6A: Logic analyzer expected output interface

Diagram 4.4A shows the expected output of logic analyzer. The user interface has 7 buttons that consist of certain functions. The purple fonts show where the buttons function. The functions are as below:

a) R1 and R2

R1 and R2 is used to let user select 2 timing. When R1 and R2 is pressed, the first

and second timing cursor will occur. The timing cursor can be moved using mouse.

b) Duration

After selecting 2 timing using R1 and R2, user can see the duration between 2 timing by pressing Duration buttons. A value and a horizontal arrow will come out between the 2 timing. The value indicates the duration between 2 timing.

c) Ruler

When pressing the Ruler button, a dotted line will come out. The dotted line can be moved by using mouse. As the button name indicate, it is used as a ruler.

d) Flatten

The Flatten button is used to display the input in square waveform. There are 8 pins, Pin 0 until Pin 7. The parallel port pin which is "on" will displays high waveform in 1 micro second. Otherwise, it will remains low waveform.

e) Bus

Bus button is used to display input in bus format.

f) Binary

Binary button is used to display input in binary format. It will display 8 bits data in 1 second.

g) Close

When user presses the Close button, the interface will close.

h) Clear

Clear button is used to clear all the data inside the interface.

B) SIGNAL GENERATOR

1. Pins interface

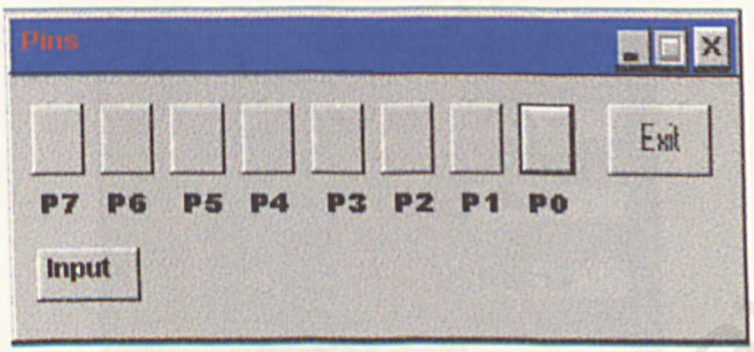


Diagram 4.6B: Pins window

This interface shows 8 pins of parallel port which are shown on pins interface. The buttons functions are as below:

a) P0 until P7 buttons

- represent 8 data pins of parallel port
- when press on the button, the button will changes color
- its function is to let user selects a pin

b) Input button

- when press on Strobe button, another interface call Input interface will come out

a) Exit button

- will close the Pins interface

2. Input interface

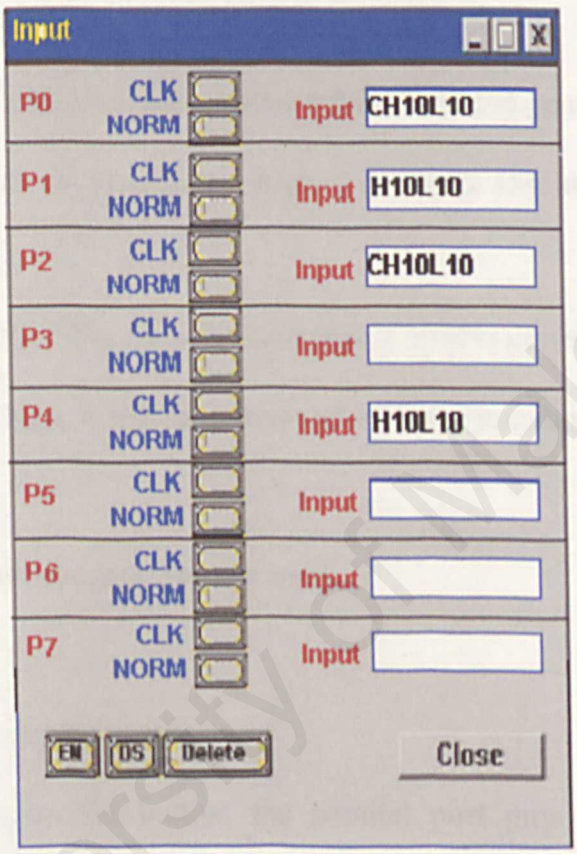


Diagram 4.6C: Stimulator window

The Input interface is used to specify clock and pins status. The buttons functions are as follow:

a) P0 until P7

- represent data pins of parallel port

b) CLK

- will presents clock waveform
- color will changes when press on it

c) NORM

- will presents normal waveform
- color will changes when press on it

d) Input

- area that is used to input clock or normal formula
- the input CH10L10 is the command that is used to generate clock waveform
- C indicates clock, H indicates high, L indicates low and 10 is the number of cycle
- the input H10L10 is the command that is used to generate normal waveform
- H indicates high, L indicates low and 10 is the number of cycle

e) Delete

- will delete all the data input by user

f) EN

- is used to refresh the pins.
- when press on En button, the parallel port pins will change its status according to latest data input

g) DS

- is used to set all the pins status to 0

h) Close

- when press on Close button, the Input window will closes

Expected outputs is shown as follow:

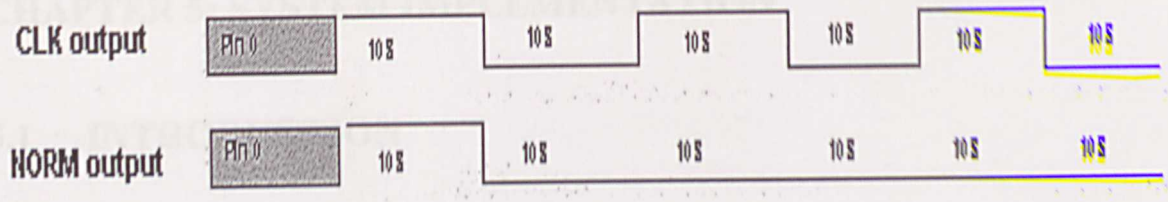


Diagram 4.6D: Expected output

4.7 SUMMARY

This chapter describes all the process and design of the proposed system and also show how data is transfer. Together with the interface design, this chapter aims to present a better understanding of the system to be developed. The next chapter will future discuss about the implementation of the system.

CHAPTER 5: SYSTEM IMPLEMENTATION

5.1 INTRODUCTION

After the system designing phase on how the system should be functioning, the next process will involves the implementation phase. System implementation is a process that converts the system requirements and designs into program codes. In a software project, the requirements analysis, system design and implementation phases do not have a clear boundary. Each phase tends to overlap one another. This phase at times involves some modifications to the previous design. The implementation phase is an important element especially when it involves a project developed by a team of people where integration of system, works by different people takes a huge effort.

The design phase earlier in the system life cycle is directed towards a final objective which is to translate the concept of the system into a software representation that is understood by the computer. The coding process involves transforming of the design into a programming language. The effort spent in this phase will actually determines the success of the system and ease the processes of modification, debugging, testing, verification, system integration and for future enhancement.

5.2 DEVELOPMENT ENVIRONMENT

Development environment plays a major role in determining the speed of developing the system. Using the suitable hardware and software will not only help to speed up the system development but also determine the success of the project. After implementing the system, the requirement of hardware and software that was stated in the previous two chapter (Chapter 3 System Analysis) can be finalized. The final list of the hardware and software tools is listed below.

5.2.1 ACTUAL HARDWARE REQUIREMENT

The hardware used to develop the system are as listed below:

- 200MHz Pentium Processor
- 128MB RAM
- 10 GB Hard Disk Drive
- Other standard desktop PC accessories such as keyboard, mouse and monitor
- Parallel port interface

The parallel port pins that is used in these system is 5 data pins and 5 status pins.

5.2.2 ACTUAL SOFTWARE TOOLS REQUIREMENTS

The actual software requirement for the system implementation is the same as listed in the chapter 3: system analysis:

- Windows 2000 server operating system and server platform
- Microsoft Visual C++ 6.0 to write coding of user interface and input/output operation
- Microsoft Office 2000 to write the documentations

The type of parallel port use in this system is in PS/2 mode. It was the bi-directional data port introduced on IBM's model PS/2. The bi-directional port enables a peripheral to transfer 8 bits at once to a PC. The term PS/2-type has come to refer to any parallel port that has a bi-directional data port. Byte mode is an 8-bit data transfer protocol that PS/2-type ports can use to transfer data from the peripheral to the PC.

1.2 HARDWARE DEVELOPMENT

This section will explain the development of the hardware component of the system. The hardware component acts as an interface between the parallel port from the computer to the system.

The system integrated two systems into one system, there are signal generator and logic analyzer. For signal generator, it needs to output data from the signal generator system to parallel port. The pins that can be used for output is data pins. The parallel port consists of 8 data pins. For logic analyzer, it need to receive data from parallel port to logic analyzer system, thus it need parallel port interface that can input data. The input pins are 5 status pins. To integrate these signal generator and logic analyzer system, only 5 data pins and 5 status pins will be used. There are data pins D0, D1, D2, D3 and D4 and status pins S3, S4, S5, S6 and $\overline{S7}$. The connection of the pins is shown on the diagram below:

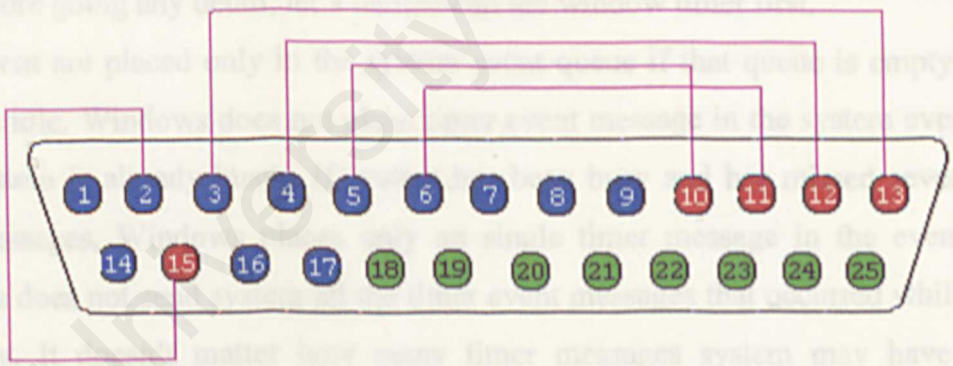


Diagram 5.3A: Male parallel port interface connection

For the purpose of externally supply power to the status pins, an external battery power supply is supplied to the status pins. The connection is shown on diagram below:

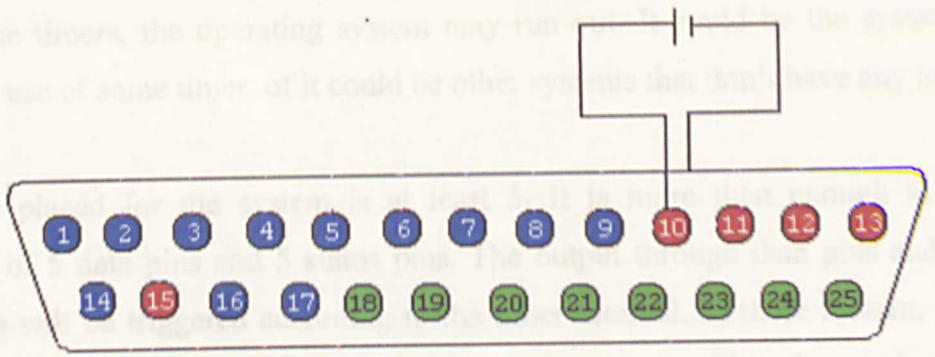


Diagram 5.3B: Connection of battery to status pin

1.3 SOFTWARE DEVELOPMENT

This section will explain the software development process.

5.4.1 TIMER

As described in the chapter 3 system analysis, the system that I build is in timer based. Thus, before going any detail, let's understand the window timer first.

Timer event are placed only in the system event queue if that queue is empty and the system is idle. Windows does not place timer event message in the system event queue if the system is already busy. If system has been busy and has missed several timer event messages, Windows places only an single timer message in the event queue. Windows does not send system all the timer event messages that occurred while system was busy. It doesn't matter how many timer messages system may have missed; Windows still places only a single timer message in system queue. [Kris Simmons, 1998]

The available range that can be set for timer in system is around 1 milliseconds on the short end to 2^32 - 1 milliseconds, or around 49 ½ days, on the long end. There are limited number of timers available to all systems in the Windows operating system. Although the number that is available should be more than sufficient for all running systems using no more than a handful of timers, if an system goes overboard and begins

hogging the timers, the operating system may run out. It could be the system that is denied the use of some timer, of it could be other systems that don't have any to use.

The timer placed for the system is at least 5. It is more than enough to perform triggering of 5 data pins and 5 status pins. The output through data pins and input to status pins will be triggered according to the timer interval. In these system, the timer interval is range from 1 milliseconds to 100000 milliseconds. Thus the maximum speed for triggering the timer is 1000 Hz.

5.4.1.1 TIMER CONTROL IN MICROSOFT VISUAL C++

To create a timer in system, basically minimum of two timers is needed. The first timer is to control the clock on the system dialog. Another timer is to let the user tune the speed, start and stop the timer as desired [Kris Simmons, 1998].

Since at least two timers is needed, thus two timer ID needed to be added to the system. Let's says the timer ID are ID_CLOCK_TIMER and ID_COUNT_TIMER. Then create the WM_TIMER function which is a build in timer function. The timer function name is OnTimer(UINT nIDEvent). Then add some code to create the timer. The code is as follow:

```
Ctime curTime = Ctime::GetCurrentTime();

Switch(nIDEvent)
{
case ID_CLOCK_TIMER:
        m_sTime.Format("%d%d%d", curTime.GetHour(),
                curTime.GetMinute(),
                curTime.GetSecond());
break;
```

```
case ID_COUNT_TIMER:
        m_iCount++;
        m_sCount.Format("%d", m_iCount);
break;
}
```

The CTime is a built in class. It is call out by using object curTime. The first case is to display the timer. The second case is to create another counter base on the timer CTime.

There are some states that the timer can do. For example, user can start the timer, stop the timer and set the timer speed. To start the timer, some code need to be added. The code is shown as follow:

```
SetTimer(ID_CLOCK_TIMER, 1000, NULL);
```

The first argument that passed to the SetTimer function is the ID for the clock timer. The second argument is how often user want to trigger the event. In this case, the clock timer event is triggered every 1000 milliseconds. The third argument is the address of an optional callback function that user can specify to bypass the WM_TIMER event. WM_TIMER is a built in timer function. The NULL argument will place the WM_TIMER event in the system queue [Kris Simmons, 1998].

To specify the speed of the timer, a variable m_iInterval is added. This variable is to let the user specifies the interval of timer that user wants. The code of start the timer according to the interval is shown as folow:

```
UpdateData(FALSE);
SetTimer(ID_COUNT_TIMER, m_iInterval, NULL);
```

The first line code is to update the data, m_iInterval input by user. The second line code is to start the timer according to the interval set by the user.

To stop the timer, the code is as follow:

KillTimer(ID_COUNT_TIMER);

5.4.2 SIGNAL GENERATOR

As described in the chapter 2 Literature review, the purpose of signal generator is to generate signal. The signal generated by the signal generator system is voltage where the system will activate the data pins by write a value 1 to the data pin. The value 1 will activate the data pin. The parallel port consists of 8 data pins. But in this system, only 5 data pins is used.

There are 3 modes of this signal generator system. There are normal mode, cycle mode and non-timing mode. The normal and cycle mode is timing based. The normal mode will generate voltage to data pin in timing based. Let's say the user generates signal for 2 seconds high and then 1 second low. The data pin will be activated for 2 seconds and then low for the rest of the time. If the cycle mode is selected, the data pin will activates the data pin to high for 2 seconds and low for 1 second. The same condition will be repeated for the rest of the time. The non-timing mode will always activate or deactivate the data pins without refer to the timing set by the user.

5.4.2.1 PARALLEL PORT WRITING DATA CODING

The following codes are written into the program to allow the user to have control over the parallel port. The code utilizes the inpout32.dll device driver to have access into the parallel port's pins. The codes are as follows:

```
typedef UINT (CALLBACK* LPFNDLLFUNC1)(INT,INT);
typedef UINT (CALLBACK* LPFNDLLFUNC2)(INT);
HINSTANCE hDLL; // Handle to DLL
```

```
LPFNDLLFUNC1 Output; // Function pointer
LPFNDLLFUNC2 Input; // Function pointer
hDLL = LoadLibrary("Inpout32");
if (hDLL != NULL)
{
    Output = (LPFNDLLFUNC1)GetProcAddress(hDLL,"Out32");
    Input = (LPFNDLLFUNC2)GetProcAddress(hDLL,"Inp32");
    if (!Output || !Input)
    {
     // handle the error FreeLibrary(hDLL);
    }
  }
Output(int, int);
```

In order to write to the data pins at the parallel port, the command used is the *Output(int, int)*, where the first *int* represents the address of the parallel port, and the second *int* represents the value of the pins.

For each parallel port, there consist of three port addresses, namely the data port, status port, and the control port. These addresses are in sequential order. For instance, if the data port is at address 0x0378, the next status port will be at 0x0379 and the control port will be at 0x037a. The addresses are in hexadecimal numbers. Thus, only the data port address is needed to write into the data pins. Because a parallel port can have different modes, namely LPT1, LPT2 or LPT3, the port address for each mode is different. In order to make sure the mode of the parallel port, the user can check the modes in the System Configuration, under the Resource Settings. But normally the parallel port will be set to LPT1 in the BIOS, and the resources settings will be from 0378 to 037F. Therefore, for the purpose of this system, the data port address will be **0x0378**.

The value in *Output(int, int)* will determine which pin to activate. Let's look at the 8 data pins as binary numbers. Each data pin can only be either enable or disable. Thus

let's assign the binary value of 1 for enable, and the binary value 0 for disable. Since there are 8 data pins, in the status where all pins are disabled the binary number will look like 00000000. Let's assume that the left most bit is pin 8 and the right most bit is pin 1. When the binary numbers are converted into decimal numbers, for 00000000 the decimal value will be 0. And for all 8 pins to be enabled the binary number will be 11111111, and the decimal value will be 255. The following shows each pin being enabled accompanied by the binary and decimal values:

- All pins disabled
  Binary: 00000000
  Decimal: **0**
- Pin 0 enabled
  Binary: 00000001
  Decimal: **1**
- Pin 1 enabled
  Binary: 00000010
  Decimal: **2**
- Pin 2 enabled
  Binary: 00000100
  Decimal: **4**
- Pin 3 enabled
  Binary: 00001000
  Decimal: **8**
- Pin 4 enabled
  Binary: 00010000
  Decimal: **16**
- Pin 5 enabled
  Binary: 00100000
  Decimal: **32**
- Pin 6 enabled

Binary: 01000000

Decimal: **64**

- Pin 7 enabled

  Binary: 10000000

  Decimal: **128**

- All pins enabled

  Binary: 11111111

  Decimal: **255**

In this system, the are only 5 data pins been selected, there are data pin 0 to data pin 4. The following table shows the function:

Table 5.4.2.1A: Command for data pins

Pin state	Decimal values	Command in codes
All disabled	0	Output(0x0378, 0);
Pin 0	1	Output(0x0378, 1);
Pin 1	2	Output(0x0378, 2);
Pin 2	4	Output(0x0378, 4);
Pin 3	8	Output(0x0378, 8);
Pin 4	16	Output(0x0378, 16);

The additional function of the system is it can generate report for the data input by the user.

5.4.3 LOGIC ANALYZER

As stated in the chapter 2 Literature review, there are two type of logic analyzer, there are timing analyzer and state analyzer. The type of logic analyzer that built here is timing analyzer but the output displayed is in binary range.

There are two ways the status pins receiving data in this system. One source is come from the signal generator system (these system integrates the signal generator and logic analyzer into one system). Another source is by externally supply the battery power (6 volts) to the status pin.

The control function of the software is to control the parallel port at the computer. The function of the program is to be able to enable and disable the status pins to receive data. There are 5 status pins of the parallel port. Enabling the receiving data from status pins is when there is a voltage present on the pins. The figure below shows the position of the status pins. The connector shown is a male connector.

Pin	Description	
1	Strobe	PC Output
2	Data 0	PC Output
3	Data 1	PC Output
4	Data 2	PC Output
5	Data 3	PC Output
6	Data 4	PC Output
7	Data 5	PC Output
8	Data 6	PC Output
9	Data 7	PC Output
10	ACK	PC Input
11	Busy	PC Input
12	Paper Empty	PC Input
13	Select	PC Input
14	Auto Feed	PC Output
15	Error	PC Input
16	Initialize Printer	PC Output
17	Select Input	PC Output

Pin Assignments

Note: 8 Data Outputs
4 Misc Other Outputs

5 Data Inputs

Note: Pins 18-25 are Ground

Diagram 5.4.3A: Pin position

There are 3 modes that the status pins will display its receiving data in the graphical user interface. There are normal, cycle and non-timing based. The normal mode and cycle mode are timing based. The normal mode receives data by showing the first state of the data input to the status pins according to the timing, for the entire time, it will display the status pin as low. Let says the user state the status pins will receives high for 2 seconds and low for 1 second, the status pins will activate the status pin to high for 2 seconds and then low for 1 second and low for the rest of the time. The output is as below:

Timer 1 2 3 4 5 6 7 8

Output ||||||||||||||||||||||||||||||_____

Diagram 5.4.3B: Output for status pin in normal mode

The cycle mode, on the other hand, will display the data by repeating the data according to timing. Let says the user state the status pin to low for 1 second and high for 1 second. The status pins will set the status pin to low for 1 second, then high for 1 second. For the rest of the time, it will repeating the same data according to timing. The output is shown as follow:

Diagram 5.4.3C: Output for status pin in cycle mode

The non-timing mode will always state the status pin to high or low only. User no need to state the timing condition.

The additional function of the system is it can generate report for the data input by the user.

5.4.3.1 PARALLEL PORT DATA RECEIVE CODING

The following codes are written into the program to allow the user to have control over the parallel port. The code utilizes the inpout32.dll device driver to have access into the parallel port's pins. The codes are as follows:

```
typedef UINT (CALLBACK* LPFNDLLFUNC1)(INT,INT);
typedef UINT (CALLBACK* LPFNDLLFUNC2)(INT);
HINSTANCE hDLL; // Handle to DLL
LPFNDLLFUNC1 Output; // Function pointer
LPFNDLLFUNC2 Input; // Function pointer
hDLL = LoadLibrary("Inpout32");
```

```
if (hDLL != NULL)
{
    Output = (LPFNDLLFUNC1)GetProcAddress(hDLL,"Out32");
    Input = (LPFNDLLFUNC2)GetProcAddress(hDLL,"Inp32");
    if (!Output || !Input)
    {
      // handle the error FreeLibrary(hDLL);
    }
  }
Input(int);
```

The command for reading data from the parallel port is the *Input(int)*, where the *int* represents the address of the parallel port.

The address of the status port is 0x039. Every time the status pins is activated, the value received by it is 1. The value received by the status pins is listed in the following table.

Table 5.4.3.1A: Command of status pins

Pin state	Decimal values	Binary values	Command in codes
Pin 0	1	0000 1000	Input(0x0378);
Pin 1	2	0001 0000	Input(0x0378);
Pin 2	4	0010 0000	Input(0x0378);
Pin 3	8	0100 0000	Input(0x0378);
Pin 4	16	1000 0000	Input(0x0378);

5.4.4 SIGNAL GENERATOR AND LOGIC ANALYZER TIMING BASED INTEGRATION

The system combines the signal generator and logic analyzer into one application. This means that the system can generate signal and receive the signal at the time. First the user set the interval value. It ranges from 1 until 100000 milliseconds. The enable button on the right hand side of the interval edit box is to used to start the timer. There are 5 areas that are display on the interface. User may selects any pin he want to input. There are high and low value which is stated in the graphical interface as "H" and "L". The "H" and "L" is let the user specify how many second they want to activated or deactivate the status pins. The data will write to data pins and send to status pins. The output will be displayed on the "Binary" column. There are multiple states that system can do. All the states are shown on the following table.

Table 5.4.4A: Software interface functions

Command button's name	Description
EnableH	Enable the pin to high without timing
EnableL	Disable the pin to low without timing
EnableC	Enable the pin to high or low based on timing in cycle mode (high first)
EnableCL	Enable the pin to high or low based on timing in cycle mode (low first)
EnableN	Enable the pin to high or low based on timing in normal mode (high first)
EnableNL	Enable the pin to high or low based on timing in normal mode (low first)
EnableEx	Allow the user to externally supply power to status pin
DisableH, DisableL,	Disable the pin and generate report for all the states

DisableC, DisableCL, DisableN, DisableNL, DisableEx	

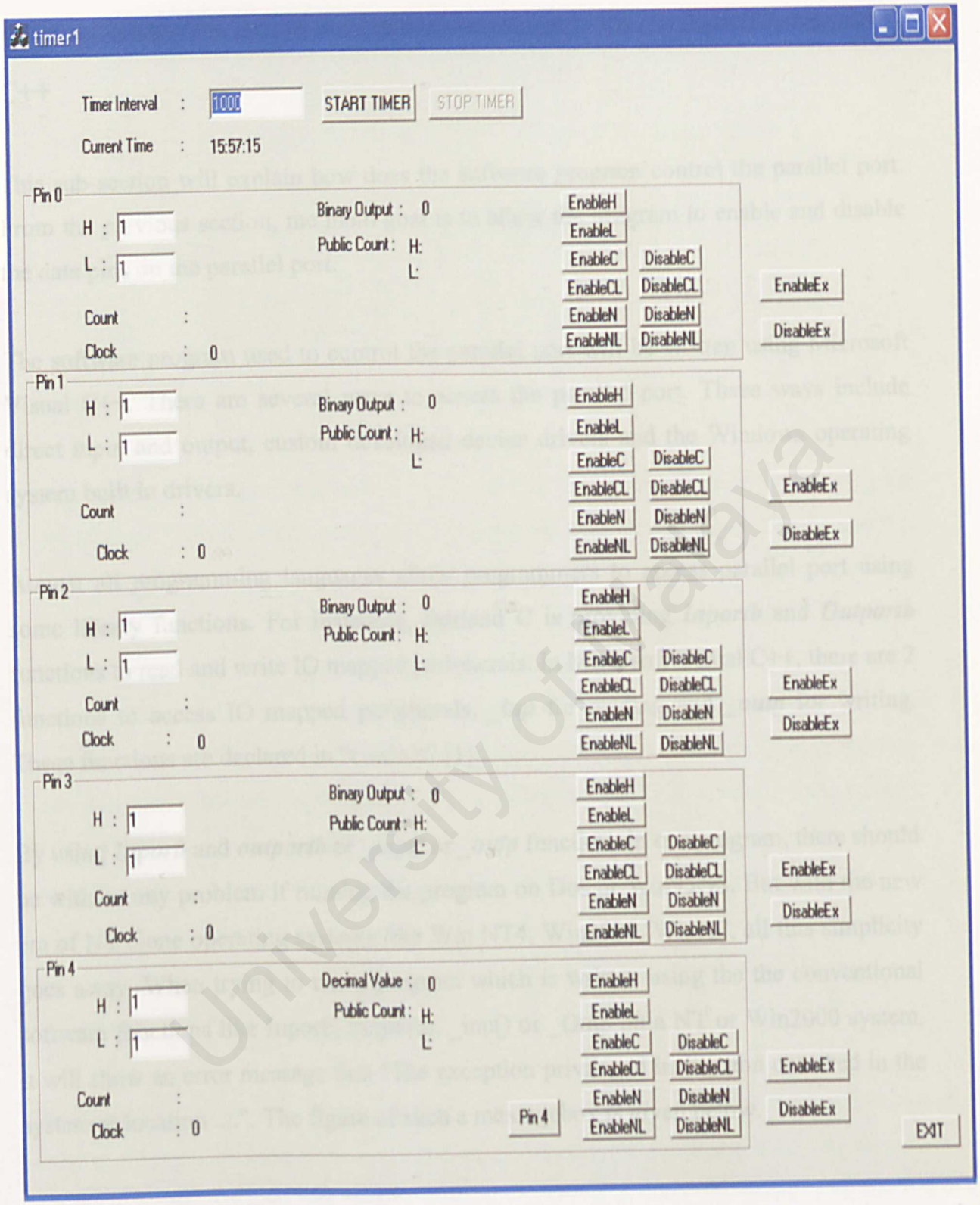


Diagram 5.4.4A: Signal generator and logic analyzer screen shot

5.4.5 PARALLEL PORT CONTROL WITH MICROSOFT VISUAL C++

This sub-section will explain how does the software program control the parallel port. From the previous section, the main goal is to allow the program to enable and disable the data pins on the parallel port.

The software program used to control the parallel port will be written using Microsoft Visual C++. There are several ways to access the parallel port. These ways include direct input and output, custom developed device drivers and the Windows operating system built in drivers.

Almost all programming languages allow programmers to access parallel port using some library functions. For instances, Borland C is providing ***Inportb*** and ***Outportb*** functions to read and write IO mapped peripherals. In Microsoft Visual C++, there are 2 functions to access IO mapped peripherals, ***_inp*** for reading and ***_outp*** for writing. These functions are declared in "conio.h" [1].

By using ***Inporb*** and ***outportb*** or ***_inp()*** or ***_outp*** functions in our program, there should be without any problem if running the program on Dos or Win95/98. But with the new era of NT clone operating systems like Win NT4, Win2000, WinXP, all this simplicity goes away. When trying to run a program which is written using the the conventional software functions like Inporb, outportb, _inp() or _Outp on a NT or Win2000 system, it will show an error message that "The exception privileged instruction occurred in the system at location ....". The figure of such a messagebox is given below.

Diagram 5.4.5A: Error Message

The above error message only happens under the NT operating system, but the program is running perfectly flawless under the Windows 98 operating system. This is because being a very secure operating system, Windows NT assigns some privileges and restrictions to different types of programs running on it. It classifies all the programs into two categories, User mode and Kernel mode, for example running in ring3 and ring0 modes. User mode programs are running in ring3 mode and Kernel mode programs are running in ring0 mode. The program that will be written falls in the user mode category. The user mode program is restricted to use certain instructions such as IN, OUT. Whenever the operating system finds that a user mode program is trying to execute such instructions, the operating system stops execution of that program and will display an error message. Eventually the interfacing program stops executing IN or OUT instructions to read or write data to parallel port. But in the same time Kernel mode program are in no way restricted in executing such instructions.

Device drivers are capable of running in kernel mode. So the workaround for the above stated problem is to write a kernel mode driver capable of reading and writing data to parallel port and let the user mode program to communicate with it [2]. The device driver is referring to the ***inpout32.dll*** for the Windows NT/2000/XP operating systems. The inpout32.dll has the following features:

- It works seamless with all versions of Windows including 98/NT/2000/XP
- It uses a kernel mode driver embedded in the dll
- No addition software or driver installation is required
- Driver will be automatically installed and configured automatically when the dll is loaded
- No special APIs are required, only 2 functions inp32 and out32
- Can be easily used with Visual Basic and Visual C++

5.4.6 MORE ON INPOUT32.DLL

The feature of Inpout32.dll is it can work with all the Windows versions without any modification in user code or the *dll* itself. The *dll* will check the operating system version when functions are called, and if the operating system is Win9X, the *dll* will use *_inp()* and *_outp* functions for reading and writing to the parallel port. If the operating system is Windows NT, 2000 or XP, it will install a kernel mode driver and talk to parallel port through that driver. The user code will not be aware of the OS version on which it is running. The dll can be used in Windows NT clone operating systems as if it is Win9X. The flow chart of the program is given below.

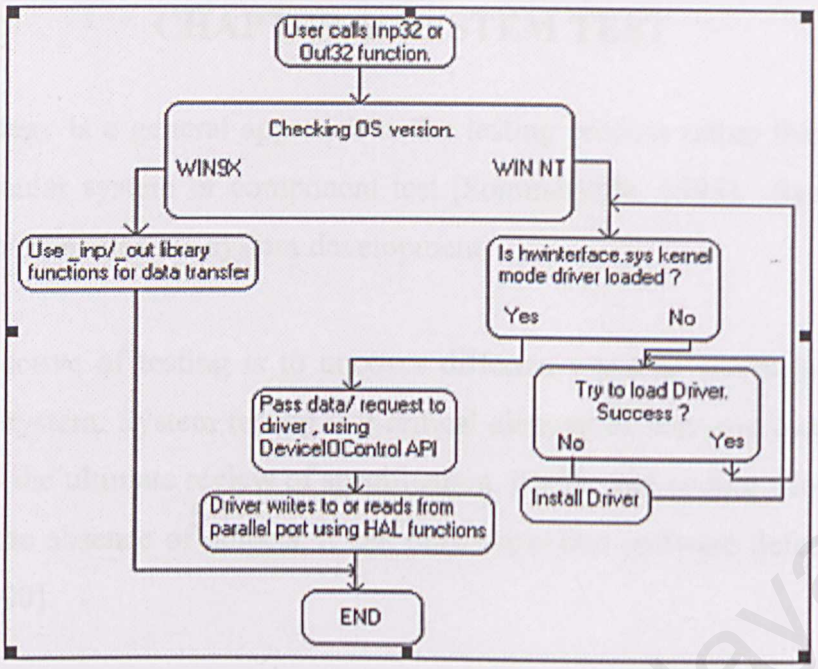


Diagram 5.4.6A: Inpout32.dll process flow chart

5.4.7 SUMMARY

This chapter explains the development process of both the software and hardware components. Besides this chapter describes the coding for both the signal generator and logic analyzer system. It explains the real software environment as well. The next chapter will further describe about the system testing use during system testing phase.

CHAPTER 6: SYSTEM TEST

A testing strategy is a general approach to the testing process rather than a method of devising particular system or component test [Sommerville, 1995]. System testing is one of the important steps in system development.

The main objective of testing is to uncover different types of errors that exist while executing the system. System testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. However, testing cannot show the absence of defects, it can only show that software defects are present [Pressman, 2000].

Testing provides a method to uncover logical error and to test the system reliability. Types of tests used are depend on what is being tested, components, group of components, or the whole system.

In developing a system, system testing usually involves several stages. First, each program component is tested on its own, isolated from the other components in the system. Such testing is known as unit testing or component testing. This stage of testing verifies that the component functions properly with the types of input and output expected from studying the component's design. After each component has been tested, the interaction between these components must be tested again to ensure that the components can be integrated.

When all components have been unit-tested, the next step is ensuring that the interfaces among the components are defined and handled properly. This step is called integration testing, also known as module testing, which verifies that the all the components work together as described in the module or system design specifications.

6.1 TEST CASE

Different test cases are applied on the system developed so that the system will be error free when the user is using it. The following are the categories of test cases being applied on the system:

- Normal data test – test by using normal data to check whether the system works properly under normal situation. For example, the number of seconds for high and low is inserted, make sure that the output come out on the binary column is correct according to the timing stated.

- Extreme data test – test with invalid data (includes input non-numerical data into a numerical field) that is not supported by the input field. For example, if 0 seconds is inserted into the interval column, make sure that an error message will come out.

- Erroneous data test – to test the performance of the system and error handling while erroneous data were input. For example, the developer will put in a value 1000 to the interval column. 1000 is 1000 milliseconds or 1 second. The speed is 1 Hz. Thus I will check whether the output is displayed in the 1 Hz speed, that is 1 seconds displays 1 value.

6.2 UNIT TESTING

Unit testing focuses on verification effort on the smallest unit of software design - software component or module. Using the component–level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovers errors is limited by the constrained scope established for unit testing. The unit test is white-box oriented, and the step can be conducted in parallel for multiple components [Pressman, 2000].

In unit testing, the following aspects are considered:

- Interface – tested to ensure that information properly flows into and out of the program unit under test. The developer has test the interface in the signal generator and logic analyzer system to make sure that data can be passed from the data pin to status pin.

- Local data structures – examined to ensure that data stored temporarily maintains its integrity during all steps in the algorithm's execution.

- Error handling path – to check whether the routines for all the error handling works properly as directed or sets. For example, the developer has added a function to ensure that field is set to character format. The developer enter data in numeric format. The error message box will prompt out and showed that the error handling works properly.

6.3 INTEGRATION TESTING

When we are satisfied that individual components are working correctly and meet out objectives, we combine them into a working system. This integration is planned and coordinated so that when a failure occurs, we have some idea of what caused it. In addition, the order in which components are tested affects our choice of test cases and tools. Some components may be in the coding phase, other may be in the unit-testing phase, and still other collections of components may be tested together.

The purpose of integration testing is to test the integration of overall performance of the system. The criteria taken in accounts are:

- Interface integrity – internal and external interfaces are tested as each modules to check if there is any lost of data across interfaces.
- Functional validity – tests designed to uncover functional errors are conducted.
- Information content – tests designed to uncover errors associated with local or global data structures are conducted.
- Performance – tests designed to verify performance bounds established during software design are conducted.

6.1 TESTING SYSTEM USING PARMON SOFTWARE

In order to produce a good test result for the software component, it is important that the process of writing the programs are planned correctly and carried out smoothly. During the writing of the program, make sure that all the required functions and methods are drafted out clearly. It is also important to frequently compiled and run the programs after each and every functions are written correctly. This way, whenever an error is found, at that moment the problem would be a small one and it should be easy to solve the problem.

The most important function on the system is to make sure that the data is write to data pin (signal generator) and is read from status pin (logic analyzer). In order to test if the output is write to the data pin and received from status pin, a third party software is used to test the parallel port's pin status. The software used is the Parallel Port Monitor written by Fred Bulback. This software is a freeware and can be download from the Internet. The Parallel Port Monitor is a utility for viewing and manipulating the state of a parallel port on a Windows operating system. The figure below shows the screen shot of the software:

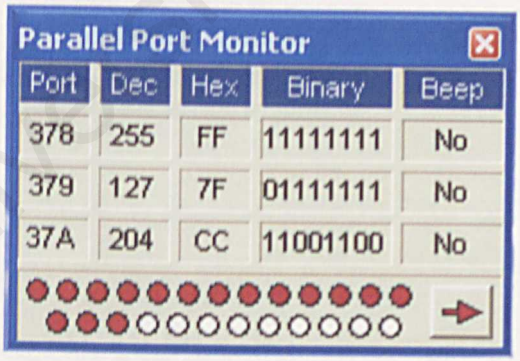


Diagram 6.4A: Parallel Port Monitor

As seen above, the coloured round dots represents the 25 pins of the parallel port. The dark red colour shows the pins that are not active, while the yellow pins are the active pins.

This software is ideally good for testing the software component of the system.

6.5 TESTING USING OSCILLOSCPE

Another way to test the data whether the data is written to the data pin and read from the status pin, a hardware oscilloscope is used. The oscilloscope is connected to the parallel port interface's pin. Every time, only 1 pin can be tested. The status of the pin will be shown on the oscilloscope screen shot.

6.6 SUMMARY

This chapter mainly explain the testing method used during the system development. The testing method are test case, unit testing, integration testing, testing system using Parmon software and testing using oscilloscope. The next chapter is the last chapter which will further evaluate the system problem, strength and so on.

CHAPTER 7: SYSTEM EVALUATION

In this chapter, the system evaluation will be discussed. After having gone through the implementation and testing phase, the final phase of developing this system is the evaluation stage.

7.1 STRENGTH SYSTEM

This system is a brand new system that integrates two applications into one system, there are signal generator and logic analyzer. Presently, there are many signal generator and logic analyzer hardware, but seldom in software. ***Thus this system proves to be unique in the sense that it combines both signal generator and logic analyzer application into one portable software.*** It can let the computer to become a signal generator and logic analyzer on the same time.

Another strength of this system is that it can generate signal and receive the signal according to timer. The output come out will displayed according to a build in timer. Such timing based signal generator and logic analyzer software does not exist yet in the previous project.

The third strength of the system is that the system can generate signal to data pins and receive the signal from status pins simultaneously. There are at least 5 timer generated in the system. Thus each pin can use its own timer to function separately. Thus at least two pins can be activated at the same time without affecting the other pins.

The fourth strength of the system is it can generate report for the data that is input by user. Every time the enable and disable function is activated, the system will write the data pin by the user into a excel file. The report will show the counter and the pins status according to the counter. Besides, if the user externally supplies a power to the status pin or deactivate the status pin, the system will detect it, receive the data and generate a report for it as well.

Besides, the system should be used for testing purpose. Hardware such as FPGA can use the system to test the data from FPGA and the output will display on the system output field.

Another strength of this system is that is can be used in smart home. Smart home is a concept that use the computerized controls in a house. For example in the computers control washing machines and microwaves, the house owner will turn the heating on and off, and they have provided new ways to monitor the safety and security of their home. Besides, the house owner can use this system to switch on or switch off their housing electronic components for example fans, lamps and so on by setting a duration into the system.

7.2 SYSTEM CONSTRAINT

There are some drawbacks of the system

- The system should have higher sampling rate to give smaller timing resolution. The sampling rate of this system is range from 0.01 kHz to 1 kHz. The maximum system speed in this system is 50 kHz.

7.3 SUGGESTIONS AND IMPROVEMENTS

- The system can be improved by creating a database. By having the database, the user will be able to trace to previous records, display more than 1 input or output data in one report.

- The system should be display in a real time graph (timing analyzer), not only in binary format. A timing analyzer is the part of a logic analyzer that is analogous to an oscilloscope. Timing mode uses an internal user-defined clock for data capture. As a matter of fact, they can be thought of as close cousins. The timing analyzer displays information in the same general form as a scope, with the

horizontal axis representing time and the vertical axis as logic levels of high and low. Because the waveforms on both instruments are time-dependent, the displays are said to be in the "time domain". Timing mode displays more information than state mode than is often necessary.

- An electronic circuit can be plug in into the parallel port to add an extra function to the system. The function is the output in the system can be displayed in a saw tooth format or sinusoidal format.

7.4 PROBLEMS FACED

Along the development process of this project, there are some problems faced:

- Software unable to run on Windows XP

  While writing the codes for the program, the platform used to compile the source code was the Windows XP operating system. The readily available functions for accessing the parallel port were unable to run correctly. This was due to security reasons on the XP operating system. Further research was done on the Internet, and a device driver was needed to overwrite the security constraint on the Windows XP operating system. The device driver used was the Inpout32.dll.

- Connecting the parallel port pin

  The connection of data pin to status pin in the parallel port interface is done by soldering the pin on the parallel port. However, the wire is easily to move out from the solder pin. Thus a parallel port socket is used. Every soldering will be done on the socket. Soldering on the socket is much more easier than soldering on the parallel port interface. Besides, it will not break parallel port interface when soldering.

7.5 CONCLUSION

This project proves to be a challenging task. This is because the system comprises of both signal generator and logic analyzer system. Apart from applying the knowledge in computer programming, a certain degree of knowledge in the manufacture and structure of parallel port is needed. This is where extra knowledge is gain and could be useful for future purposes.

Due to the problems encountered during development, it is sad to mention that the developer of the project was almost behind schedule. But with great dedication and beliefs, the developer managed at the end to complete the project.

Along this project, the developer gained many experiences. With these experiences gained, the developer hopes that all these will be put into practice when graduate from university.

7.6 SUMMARY

This chapter explain the system evaluation which involves determining the problems or difficulties, which arise during and after the program coding phase, recognizing the system strengths and weaknesses, and finally draft out the system limitations and also its future enhancements.

APPENDIX I: USER MANUAL

1. First, user has to put in the value of interval into the timer interval field. The value range from 1 until 100000 milliseconds.
2. User may click the "Start Timer" button to start the timer.
3. Next, user may select one pin from the five pins on the interface.
4. Key in the number of seconds for high and number of seconds for low into the "H" and "L" fields.
5. The "H' and "L" value will determine the timing representation on the output.
6. Now, user may click the six "Enable" buttons on the right hand side of user.
7. The "Enable" button consist of different function which is stated on chapter 5, System Implementation. User can see the selected mode on the "Binary Output" output field.
8. When user want to disable the output and timing, user may click the "Disable" button.
9. Every time user click the "Enable" and "Disable" button, a report will be generated on the excel file. If user want to read the report file, user may go the directory that contain the "Exe" file. There are 3 files for each pin, there are "RecordPin", "RecordPinPL" and "RecordPinPH". The file "RecordPin" is to generate record for "EnableC", "EnableCL", "EnableN" and "EnableNL". The file "RecordPinPL" and "RecordPinPH" is to generate the record for "EnableEx" and "DisableEx". "RecordPinPL" is record for user when they clicking the "DisableEx" button on the time the "L" value is counting. "RecordPH" is record for user when they clicking the "DisableEx" button on the time the "H" value is counting.
10. The "EnableEx" button is used when user want to externally supply a power to the status pin. Let say, user activates the status pin S3 by connecting the battery to the status pin S3. When they click "EnableEx", a counter "H" field (under "Binary Output") will start counting the duration of user activates the status pin. If the user only activate the pin for 4 seconds, the "H" will display until 4.

11. Now user has plug out the power supply. The state of the status pin is now on low condition. Thus the status pin become low. A counter will start counting the low state of status pin on the "L" field just under the "Binary Output" field. Let say user only want the low value count until 5 seconds, thus user may click the "DisableEx" button now.
12. To read the report that is just input, user may go to the "exe" file directory and open the "RecordPinPL" file.
13. The five pins can be selected to display a output on the same time.

REFERENCE

[Agilent Technologies, 2000]: "What's a logic analyzer?", Agilent Technologies, 2000 .

[Allen, 1998]: "What is VB", Allen L. Wyatt & Cavett Pease, International Info Server 4.0-Administrator's Guide, Prima Publishing, pg.224, 1998.

[Arian, 2001]: "A logic analyzer using the PC's parallel port", Arian van Dorsten .

[Bob Perrin, 2003]: "Digital Inputs", Bob Perrin, Circuit Cellar Online, ChipCenter Questlink, 2003 .

[Craig Maynard, 2000]: "Logic Analyzer Operation", Craig Maynard, 2000 .

[Dage, 1999]: "Using the Parallel Port", Dage Scientific, 1999 .

[ELEC 2010, 2002]: "Experiment 2 Oscilloscope and Function Generator", ELEC 2010, Experiment 2 PRELAB, 2002.

[Jan Axelson, 1999]: "Parallel Port Complete, programming, interfacing & using the PC's parallel printer port", Lakeview Research, Madison, Pg 1-11, 17-22, 129-148 & 149-164, 1999.

[John B, 2003]: "IO.dll", Geek Hideout,2003 http://www.geekhideout.com/iodll.shtml

[Kris Simmons, 1998]: "Sams Teach Yourself Visual C++ in 21 Days", Second Edition, Sams Publishing, pg 67-104, 1998.

[Kyle C, 1999]: "Building an 8-bit PC-Based Logic Analyzer", Kyle C. Quinnell, Department of Engineering Technology New Mexico State University, 1999 .

[Martin Clausen,2002]: "Digital Signal Generator", Martin Clausen, 2002.

[Mautin, 2001]: "Daqarta-Stim3a advanced stimulus signal generator", Mautin, Interstellar Research, 2001 .

[Mohammed Elzubeir, 2000]: "Parallel Port Programming", Mohammed Elzubeir.

[NI, 2003]: "Using the Parallel Port as an Input/Output Channel", National Instruments, 2003 .

[Nick, 2000]: "C++ programmers to benefit from studying the manuals", Nick Langley, Microsoft, Technology, 2000, Thursday 3 August 2000 .

[Peacock, 2001]: "Introduction to Parallel Ports", Craig Peacock, 19th August 2001 .

[Peacock, 2003]: "Interfacing the Enhanced Parallel Port", Craig Peacock, 2003 {http://www.beyondlogic.org}.

[Shauna Rae, 1999]: "Getting an ADC0809 Analog to Digital Converter to Work for you", Shauna Rae, National DataSheet, National Application Notes, Texas Instruments Datasheet,1999.

[Tapcott, 1999]: "Creating Value in the Network Economy", Tapscott, Don. (Ed.), President and Fellows of Harvard College, 1999.

[Tomi, 1996]: "Simple circuit and program to show how to use PC parallel port output capabilities", Tomi Engdahl, 1996-2000

[Whitten, 2000]: "System analysis and design methods, Whitten, J.L., Bentley, L.D., & Dittman, K.C. , McGraw-Hill, 2000.