IJCCCE, VOL.7, NO.2, 2007

DEVELOPMENT OF PC BASED MULTI-CHANNEL PROGRAMMABLE LOGIC ANALYZER

Dr. Ali A. Ati, M.Sc. Eng. Sarmad Hassan Ahmed

Receipted: 23/11 /2005
Accepted:10/ 2/ 2008

Abstract

This paper presents the design and practical implementation of a multi-channel PC based logic analyzer. The analyzer has16 input channels with memory depth of 4K snapshots/channel and capture rate of up to 5 MHz. The analyzer parameters such as internal or external trigger source, falling or rising edge capture clock, state or timing measuring mode, number of pre-trigger and post-trigger snapshots, are made to be programmable and could be changed manually. The analyzer prototype consists of hardware part represented by the development of the interface ISA card and the software part that involves the kernel mode driver and the GUI program development. The developed prototype analyzer has been tested under different configuration schemes using 8085 SDK.

الخلاصة

يقدم هذا البحث التصميم و التنفيذ العملي لمحلل منطقي متعدد القنوات باستخدام الحاسوب الشخصي. الجهاز المصمم يحتوي على 16 قناة إضافة إلى سعة خزن 4K snapshots/channel ومعدل اخذ إشارات يصل الى 5MHz . خلال التصميم تم مراعاة ان يكون الجهاز قابل لتغير خواصه من قبل المستخدم ، هذه الخواص تتضمن طبيعة مصدر التحفز داخلي أم خارجي ، نوع طور التشغيل ، وكذلك عدد الإشارات المأخوذة قبل وبعد حصول التحفز .

تنفيذ الجهاز تضمن جزئيين رئيسيين : الأول هو الجزء البنيوي الذي يقوم بعملية اخذ الإشارات من القنوات وخزنها في الذاكرة الموجودة في الجزء البنيوي نفسه ،. أما الجزء الثاني فهو الجزء البر مجي ويتكون من سواقة وواجه تطبيق للمستخدم تمكنه من استخدام الجزء البنيوي والسيطرة عليه، وكذلك عرض الإشارات المستلمة بطريقة ملائمة . تم اختبار النظام في مختلف طرق عمله وخواصه من خلال استخدام منظومة المعالج الدقيق 8085-SDK kit.

Al-Nahrain University, College of Engineering, Computer Engineering Department
E-mail: abudinoor@yahoo.com

List of Abbreviations

TTL	Transistor-Transistor Logic
CMOS	Complementary Meta Oxide Semiconductor
ISA	Industry Standard Architecture
IOR	Input Output Read
IOW	Input Output Write
I/O	Input/Output
SRAM	Static Random Access Memory
CS	Chip Select
WR	Write Signal
OE	Output Enable Signal
CPU	Central Processing Unit
ISR	Interrupt Service Routine
API	Application Programming Interface
GUI	Graphical User Interface
SDK	Software Development Kit
ALE	Address Latch Enable
FPGA	Field Programmable Gate Array
PCI	Peripheral Component Interconnect

1. Introduction

The PC based type of logic analyzers has more flexibility than stand alone logic analyzers. Very limited papers have been published in this filed. Anderw March [1], presents a 32 channel, 40 MHz, fully PC controlled TTL/CMOS logic analyzer with internal/external triggering and trigger delay. The presented prototype is little bit complicated and has short flexibility. The parameters of the analyzer are not programmable and need much effort to be changed. Kyle C. Quinnell [2], presents a way of using a PC computer as an 8-bit logic analyzer. The technique uses the capabilities of the PC computer's parallel port to provide an 8-bit input. The software that provides the interface was written using Turbo C++. No interface card was used because it depends completely on the parallel port of the PC. The biggest drawback with this prototype is the low capturing rate. This paper presents the development of a PC based multi-channel programmable logic analyzer. Section two of this paper outlines the general block diagram of the proposed logic analyzer and the circuit diagram with operation of the hardware part. Section three presents the software part details while section four shows the test results for the analyzer prototype. The last section, section five, presents the conclusions and suggestions for future work.

2. Hardware part

The general structure block diagram of the proposed logic analyzer is shown in Figure (1). This block diagram contains the major functional blocks of the designed card, and shows the interconnections between them, which gives an idea about the data flow inside the system. The Orcad simulation program has been used as a schematic design entry tool as well as to verify through simulation the operation of the circuit before practical implementation. Below is a description of each element:

2.1 ISA Bus Interface

Buffers are used to isolate the ISA interface bus from the internal circuit of the add-on card [3]. Buffering is accomplished by the 74LS244 for unidirectional transfer and by 74LS245 for bidirectional transfer. In the latter, an additional signal is needed to specify the direction of transfer, which is derived from the -IOR and -IOW signals.

2.2 Decoder Circuit

The decoder circuit decodes the addresses of the I/O registers needed by the add-on card by selecting a specific address from a given range of addresses and outputting a signal indicating that the address was found on the address bus. The design used the address range from 300h to 306h on even addresses only. Even addresses are used because 16-bit I/O registers are being interfaced only. Figure (2) shows the full circuit diagram of data buffers and decoder circuit. 3-to-8 decoders (74LS138) are used widely for address decoding. In this figure U6 and U8 are used to decode the base address which is 300h. While U4 and U5 decode the even addresses in the I/O range, and separate the Input registers from output registers with the use of –IOW and –IOR signals. Figure (3) shows the timing diagram of the decoder circuit for the address 300h. That was a result of requesting a word input from an input device through the command “inpw (300h)”.

2.3 Command & Status Registers (Control Registers)

Output data from the computer are the configurations’ commands. These commands must be stored in the command registers. Those registers are latches (74LS374) to store the data from the data bus before they vanish. After the data configurations are correctly latched from the data bus, the add-on card can use them whenever it is needed. Therefore the purpose behind using those registers is to store the configurations set by the software directly on the electronic board from which it can be read. Input registers are used to put the data on the data bus when the computer requests it through the inpw command. Input registers are called status registers. Figure (4) shows the command and status registers circuit diagram with the connection between each register and the decoder. The decoder directs the appropriate signal to the register when it is accessed. Each bit of a command or a status register has a specific function, which is denoted by Figure (3).

2.4 Main Analyzer's Circuit

All the circuits above are used to prepare signals and configurations that are used by the main analyzer’s circuit. This circuit can be subdivided into parts as follows:

2.4.1 Capture and Storage Circuit

This part is responsible of capturing snapshots and storing them in a temporary buffer on the add-on card.

The capture process is synchronized with a clock introduced by the clock circuit part mentioned latter. The circuit diagram is shown in Figure (5). The Latches 74LS374 take snapshots with the clock transition from low to high then they are copied to the memory buffer. The memory buffer is a 4K volatile SRAM (part number is HM6264-10) with an access time of 100ns. This buffer has to record the snapshots and then forward it to the PC when all the snapshots required were taken and stored. Thus it must be connected to the latches mentioned above in addition to the ISA data bus. In order to isolate those data paths an extra 3-state buffer must be used to separate data coming from the latches and the data directed to the data bus of the PC. Timing diagram for both cases are shown in Figure (6.a) and Figure (6.b) respectively. To write the snapshots to the memory buffer, it is important to prepare two signals. The first one is called Chip Select signal (-CS) and the other is called Write signal (-WR). The write operation begins with at the point where both of these signals go low, and finished whenever anyone of them goes high. A time of 100ns must be guaranteed in order to ensure the proper storage of data A read operation is made when the –WR signal is high and –CS is low with the Output Enable signal (-OE) goes low.

2.4.2 Memory Address Counter

The memory must be supplied by an address in which the snapshot is going to be stored (or going to be read from). This address is to be generated by Memory Address Counter. The counter must be incremented with every sampling clock. Figure (7) shows the circuit diagram of the address counter that consists of 3 cascaded 4-bit counter of type 74LS191.

The outputs of the Memory Address Counter are connected directly to the address pins of the Static RAM in the previous circuit. The time needed for a single snapshot to be taken is estimated by the access time required for the SRAM to store data (SRAM access time) plus the time required for the counter to increment itself. Figure (8) shows the timing diagram for the cascaded counters.

2.4.3 Control Circuit

Control circuit is where the decisions about starting and stopping counting snapshots take place. It also prepares signals for memory buffers, counters, interrupt circuit, clock and internal trigger circuit. Figure (7) above, shows the circuit diagram for the control circuit.

When the system starts its operation, it begins by taking snapshots and filling the circular buffer. At the time the trigger comes to the flip-flop, it changes its state (Q output) from high logic to low. This transition activates the snapshot counter, which will go decreasing its count with every clock pulse. The counter must be loaded with the number of post-trigger samples before the system starts its operation or it will never start. When all the required snapshots are stored in the memory buffer, the flip-flop will automatically disable the counter and it will then stop further storage of snapshots.

2.4.4 Snapshots Counter

This counter sets the number of snapshots to be taken directly after the trigger comes to the control circuit. The analyzer will start to capture snapshots as soon as the counter contains a number other than Zero. Setting the counter with the number zero will cause the NAND gate shown in Figure (7) to by pass a signal to the control circuit indicating that the terminal count has been reached. The latter in turn, stops capturing snapshots since the samples required have all been taken.

2.4.5 Interrupt Circuit

Figure (7) also shows the interrupt circuit. The Computer CPU is interrupted whenever the analyzer finishes taking captures. The low to high transition in the control circuit flip-flop strobes the high logic into the output of the interrupt flip-flop which is connected to the interrupt request line of the ISA bus. The interrupt flip-flop is cleared at the beginning of the Interrupt Service Routine (ISR). This is accomplished by connecting the enable of the (302h) input register from the decoder to the clear pin of the flip-flop.

2.4.6 Clock Circuit

Snapshots are taken whenever a transition from low to high takes place on the clock signal. Clock can be programmed as internal or external. External clock comes directly from outside the analyzer i.e. from an external source usually the system clock of the device under test. External clock is used in state measurement only. Internal clock is used in timing measurements and it is initiated onboard. Figure (9) shows the circuit diagram of clock circuit. The rate divider 74LS294 is used to provide programmable clock frequency from the 4MHz system clock provided by the ISA bus. These dividers provide a division capability from 2^1 to 2^{31} [4]. The clock circuit is also responsible for the programmable clock sampling edge feature. The XOR gate is used as an inverter depending on the value of the command bit associated with configuring this feature.

2.4.7 Trigger Circuit

Figure (9), shows the circuit diagram of the trigger circuit. The two 8-bit comparators (74LS688) are used in pattern comparison for pattern based triggering. The output of the 8-bit comparator is low when both input patterns are exactly the same. Pattern triggering (internal trigger) takes place in five modes. These modes represent the state at which the two different comparators have to be in order to make a trigger. The first mode is when the outputs of both comparators are low at the same time. And the second mode is when the output of one of the comparators is low. Third mode is activated either when one of the comparators' output is active. The forth and fifth modes make a trigger depending on a single comparator only. Mode selection is made through programming the command register with the appropriate multiplexer selection bits. Only one mode can be selected at a time. For the external trigger, this circuit provides capability to choose the transition type of the trigger signal on which the triggering of the analyzer happens. The XOR gate connected to the specific bit in the command register of the analyzer accomplishes this feature.

3. Software part

There are many available platforms on which the software can be based. The software developed for the logic analyzer is based on Microsoft Windows, which is the most widely used platform by personal computers. The software design is consists of two main parts: 1) Kernel-mode driver 2) User-mode application program [5]. The software parts developed in this work will be fully compatible with all versions of Microsoft Windows except Windows95, due to limitation in this old version of Windows operating system. Software development in MS Windows is largely dependent on the factory ready routines, called API, supplied by Microsoft Corporation. These routines have a standard call parameters and outputs [6]. Two steps are involved when writing software for a hardware board. The first step is preparing interface routines for introducing functions available by the hardware board and those routines are called 'Driver'. The Drivers works in conjunction with Windows to process interrupts, and carry out I/O operations for a given application without disrupting the execution of other applications. The other step is designing and providing function for the graphic user interface program (GUI) which the ordinary user is supposed to interact with. The GUI program uses the routines prepared by the Driver to communicate with the hardware board indirectly. Program developed in this step represents the visual part of the whole software to support the hardware board.

3.1 Driver Routines

Generally, in order to do its job the driver consists of many routines, some of them are standard routines, while the others are optional and exist according to the functions carried by the hardware card that the driver has to manage [7]. The developed driver has the following routines:

1- DriverEntry Routine
2- AddDevice Routine
3- Dispatch I/O Routine
4- StartIo Routine
5- IoCompletion Routine
6- IoConnectInterrupt Routine
7- KeSynchronizeExecution Routine
8- Interrupt Service Routine (ISR)
9- DpcForIsr Routine
10- Unload Routine

The driver developer has to prepare routines to interact with the hardware board I/O ports and the capability of Interrupt handling. Ports used by the logic analyzer add-on card are shown below:

I-Input ports:

300h:Snapshot register.

For reading captured snapshots from the memory on the card.

Program uses this port to retrieve the snapshots stored in the temporary memory of the card.

302h:Trigger Address register.

Stores the address of the snapshot at which the trigger has taken place.

304h: Status register.

It is used to acquire the current working status of the analyzer.

II-Output ports:

300h:Pre-trigger snapshots.

This port is used for setting the number of snapshots needed to be stored after the trigger arrival. Storing a value other than zero in this port will cause the start of a new test.

302h Clock Divider register.

This register is used to hold the rate by which the ISA clock is to be divided by to obtain the capturing clock required for timing analysis measurement.

304h: Command register.
Used to configure the analyzer's clock source, trigger source, pattern trigger type and active clock and trigger edges.

306h: Trigger pattern bits.
Pattern trigger bits are set using this port. To use pattern triggering the appropriate bits in the Command Register must be set also.

Each one of these I/O Port has to be implemented in special routine called Dispatch I/O Routine. Figures (10) to (11) show the development of some mentioned routines. For IoConnectInterrupt and KeSynchronizeExecution routines the predefined standard routines were used.

3.2 Graphical User Interface (GUI) Development

The graphical program interface has been designed in Visual Basic Language to look as in Figure (12). Configuration for the analyzer must be completed before clicking on the Start Button. Configuration involves setting pre-trigger number of samples, clock source, trigger source …etc. Clicking on the start button makes the program goes into the standby state waiting for the analyzer interrupt to acknowledge availability of data. When the snapshots are complete, an interrupt occurs telling the program to fetch the data from the hardware. The snapshots are then displayed as waveforms on the display window. The program menu shown in Figure (12) has the following keys:

Load: To load a previously saved snapshots file and display it as a waveform.

Save: Saves the captured snapshots into a file.

Parameters: shown in Figure (13). From this window the following parameters are determined. (Post-trigger snapshots, analysis type, trigger source, clock edge type, trigger edge type, and trigger mode).

Start: To begin taking snapshots according to the previously set parameters.

Abort: To stop the snapshots capturing process.

4. Experimental Test.

The overall developed system was tested using the 8085 SDK. A small program was written as shown in table (1) to initiate predefined transactions on the address, data, and control bus. Accordingly, the system functionality can be tested by comparing captured snapshots by the system with the theoretical bus timings. The functionality of the logic analyzer was tested in 6 different tests as illustrated in table (2) and the logic analyzer channels during all tests were connected as outlined in table (3). Note that, only the low byte of address lines was connected to the analyzer. The six tests mentioned before, were examined by connecting an 8085 SDK to a PC on which the developed hardware is connected, and with its driver and GUI programs are being executed. Because of space limitations, the results for two of these tests are shown and discussed below.

- **Test 1**

Figure (10) shows the output screen of the 12 channel connected in test 1. The changes of signals are not clearly visible, therefore the zooming capability in the program was used to get a closer view for the signals. The numbers on the axis represent the snapshot number. Pre-trigger snapshots (numbers with –ve sign) lies before the snapshot instance (origin point). Post-trigger snapshots lies after the snapshot instance (numbers with +ve sign). The y-axis represents the channels, and the panel on the right shows what each channel is connected to, (the panel text is predefined). As mentioned in [8], the fetch cycle is composed of four T states. The first T state can be considered as the address phase, where the rest are considered as data phase. In Figure (10) the two phases of the (DCR C) instruction are being pointed to. For the address phase, the low byte of 2002h is shown (which is 02h) and for the data phase, the opcode (0Dh) is shown afterwards. Since this test was done for the rising edge, the ALE signal activation could not be captured because the ALE is activated during the low part of the CPU clock only.

- **Test 2**

In test 1 the fetch cycle was discussed only. In this test the fetch and memory read cycles are going to be discussed. In figure (11), the two cycles of the (MVI C 0Fh) instruction are being pointed to. Since this test was done with the falling edge capture clock, the ALE signal activation could be captured. In the address phase of the fetch cycle the low byte of 2000h is (00h). The Address Phase can be distinguished by the ALE signal being high, while in the Data Phase the ALE goes low. The data phase of the fetch cycle is the opcode (0Eh) as it is shown by the figure. In the address phase of the memory read cycle the low byte of 2001h is (01h) and in the data phase of the memory read cycle the data byte is (0Fh) as shown by the figure too.

5. Conclusion and future work.

1- The maximum capture rate achieved was 5MHz, because of the limits concerning components' speed and the wire wrapping connection method used. This rate can be increased by many ways, for example, building the circuit as Printed Circuit Board PCB, using faster components, using programmable logic devices (to minimize the number of components and thus minimizing signal delay), and also using fast RAM to store the captured snapshots.

2- The use of 16 bit interfacing with ISA bus produced speed up factor for the system performance, instead of taking 8-bit in two stages, which increases CPU clocks required for a single Input/Output instruction and thus it increases the associated delay.

3- Drivers under windows represent a successful method to interact with the developed hardware in MS Windows Operating System, which gives the capability of developing a friendly GUI. It also enlarges the way toward more modular and configurable software implementation for the system.

4- Experimental results reflect that the designed and implemented logic analyzer system operates properly as they were compared to the corresponding theoretical waveform diagrams.

And a suggested recommendations for future work is summarized as:

1- Implementation of an FPGA based logic analyzer. In order to improve the analyzer specification, an FPGA chip can be used for increasing memory depth, (so that the system would be able to capture more snapshots and analyze more transactions), speeding up capture rate (to make the system capable of debugging faster systems, or to increase the resolution of time measurements), and increasing the number of channels (to make the system capable of debugging more IO-signals systems).

2- Upgrade the design to be interfaced to the PCI Bus. In order to be compatible with the new computer systems. Also this would increase the bits being transferred at each transfer operation, decreasing by this the CPU engagement time especially if the burst transfer mode is used. Interfacing the developed system to the PCI would further provide system portability, and dynamic resources allocation through the Plug'n'Play feature.

References

[1] Andrew March, "PC Based 32 Channel Logic Analyzer", Electronics Australia in Oct/Nov, 1996.

[2] Kyle C. Quinnell, "Building an 8-bit PC-Based Logic Analyzer", Department of Engineering Technology, New Mexico State University, 2001.

[3] William Buchanan and Austin Wilson *Advanced Pc Architecture,* Addison-Wesley, 2001.

[4] Walter A. Triebel and Prentice hall, *The 80386, 80486, and Pentium processor hardware, software and interfacing*, Prentice-Hall Inc., 1998.

[5] Matt Pietrek, *Windows Programming Secrets*, IDG books worldwide Inc, 1995.

[6] Herbert Schildt, *Windows 98 programming from the ground up,* McGraw-Hill publishing company, U.S.A., 1998.

[7] Microsoft Corporation, "Windows Driver Model (WDM) Technology", available at http://www.microsoft.com/hwdev//driver/wdm/default.asp#top

[8] Chares E. Stewart, The Intel Microprocessors Architecture, *Programming, and Interfacing,* Fifth edition, Prentice-Hall Inc., 2000.

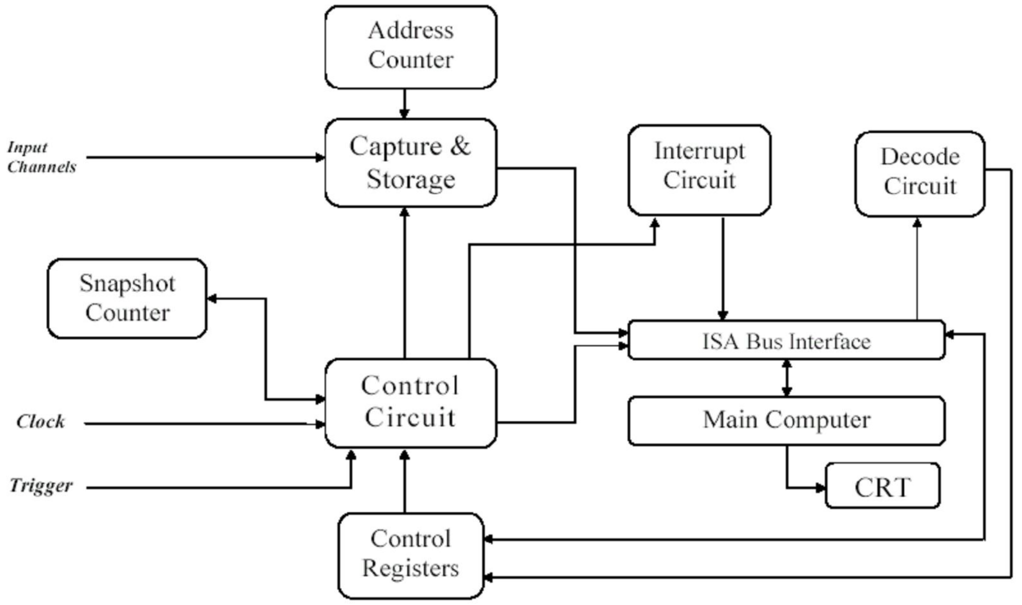


Figure (1): Structure block diagram of the logic analyzer.

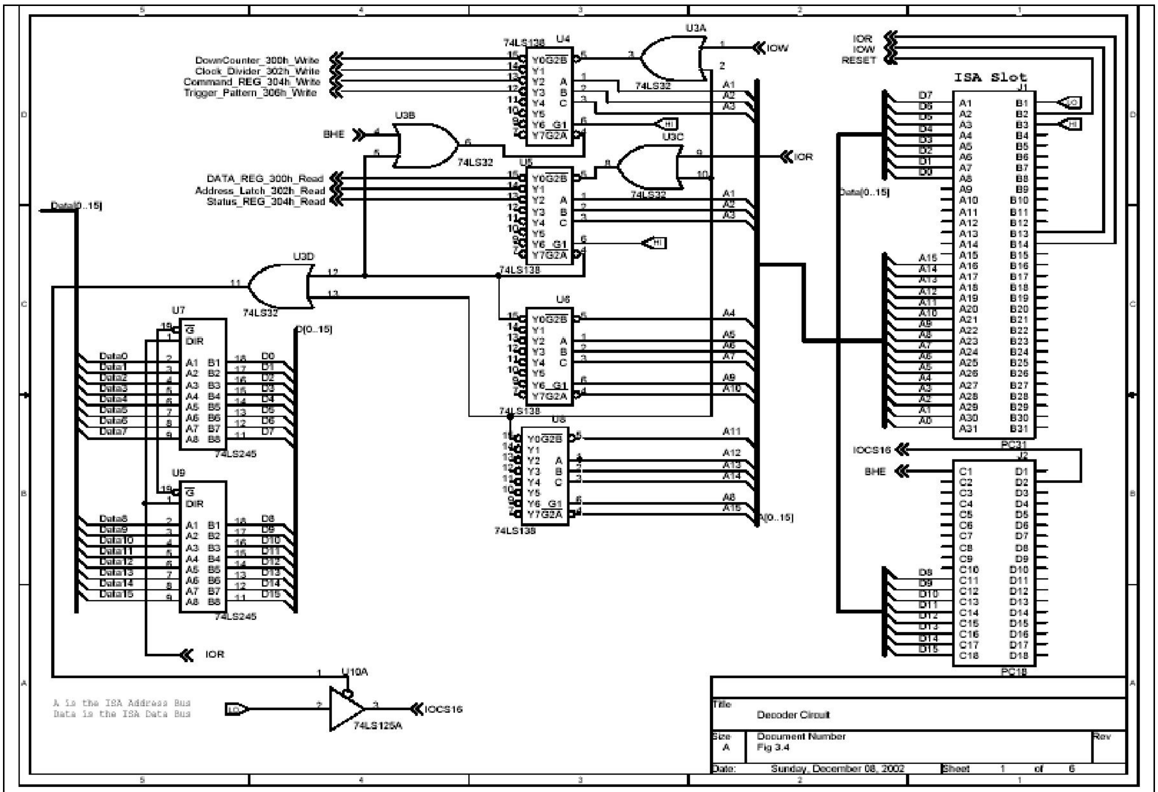
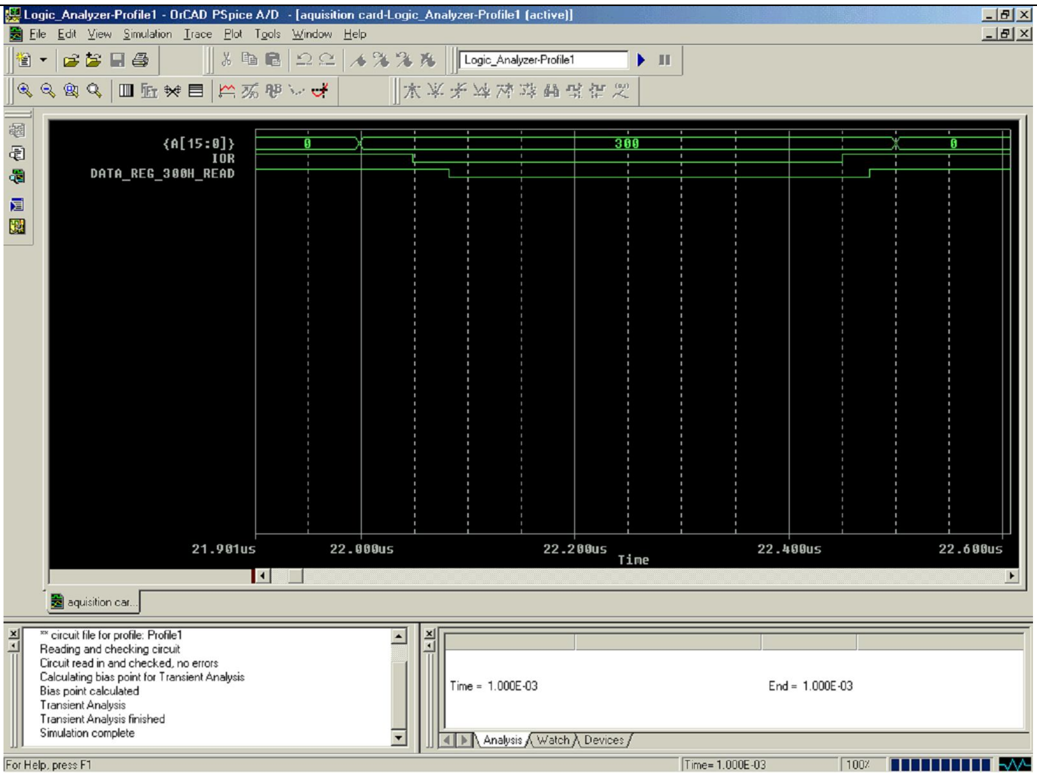


Figure (2): Data Buffers and Decoders circuit.


Logic_Analyzer-Profile1 - OrCAD PSpice A/D - [aquisition card-Logic_Analyzer-Profile1 (active)]
File Edit View Simulation Trace Plot Tools Window Help
Logic_Analyzer-Profile1
{A[15:0]}
IOR
DATA_REG_300H_READ
0
300
0
21.901us
22.000us
22.200us
22.400us
22.600us
Time
aquisition car...
Analysis
Watch
Devices

** circuit file for profile: Profile1
Reading and checking circuit
Circuit read in and checked, no errors
Calculating bias point for Transient Analysis
Bias point calculated
Transient Analysis
Transient Analysis finished
Simulation complete

Time = 1.000E-03

End = 1.000E-03

For Help, press F1

Time= 1.000E-03

100%

Figure (3) Decoder circuit simulation result for the address 300h.

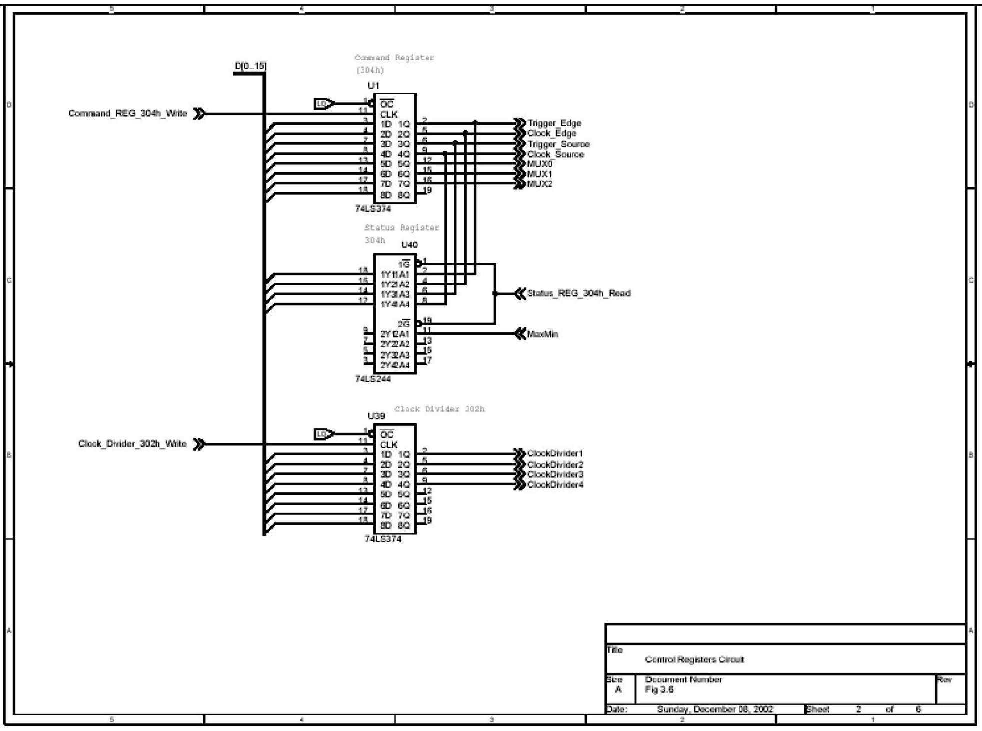


Figure (4): Control registers circuit.

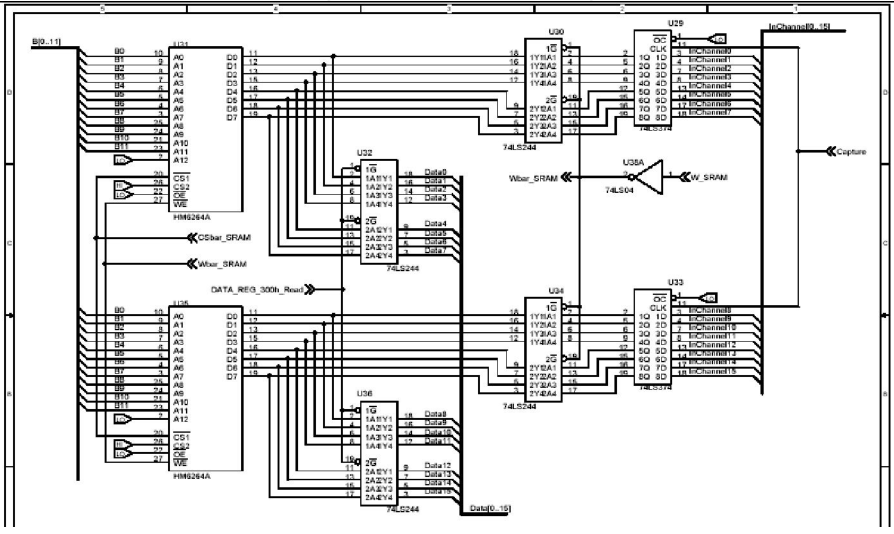


Figure (5): Capture and Storage circuit..

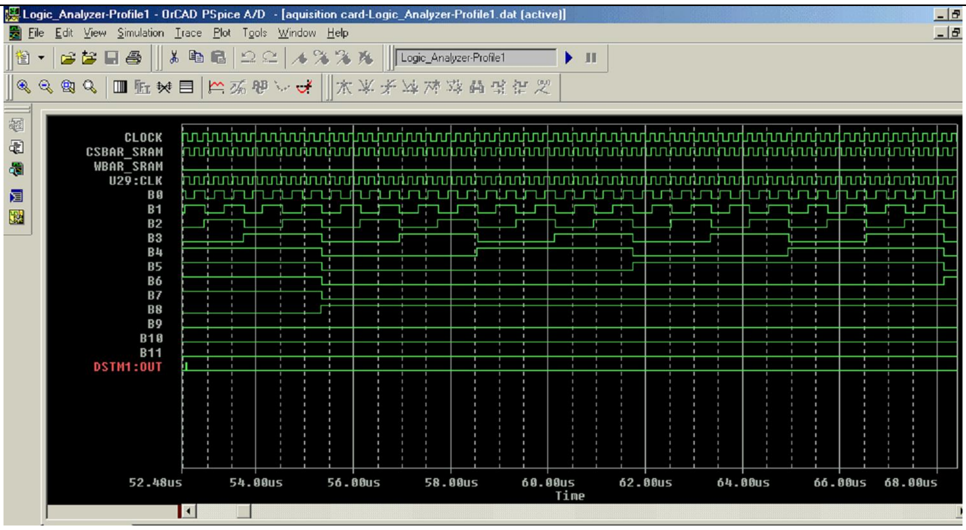


Figure (6-a): Recording phase timing diagram.

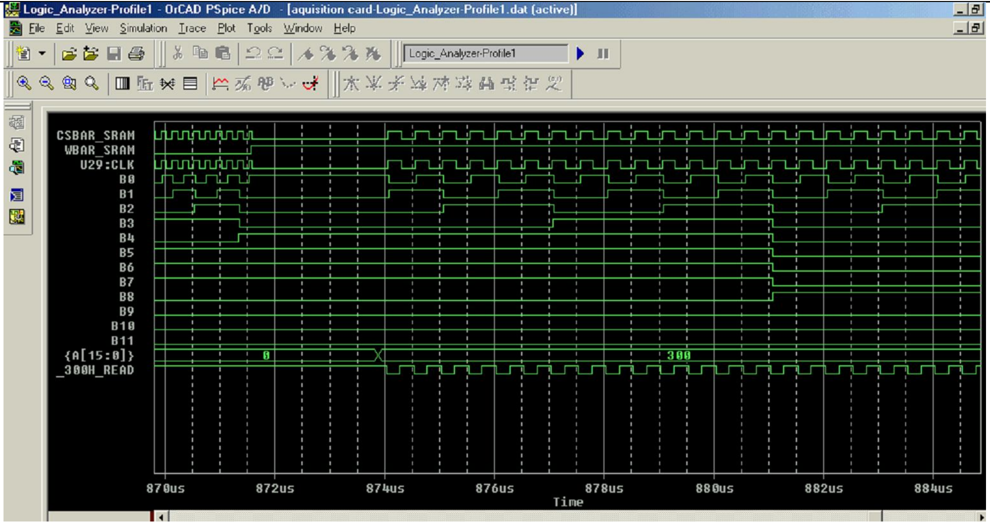

Logic_Analyzer-Profile1 - OrCAD PSpice A/D - [aquisition card-Logic_Analyzer-Profile1.dat (active)]
File Edit View Simulation Trace Plot Tools Window Help
Logic_Analyzer-Profile1
CSBAR_SRAM
WBAR_SRAM
U29:CLK
B0
B1
B2
B3
B4
B5
B6
B7
B8
B9
B10
B11
{A[15:0]}
_300H_READ
0
300
870us
872us
874us
876us
878us
880us
882us
884us
Time

Figure (6-b): Storing into PC timing diagram.

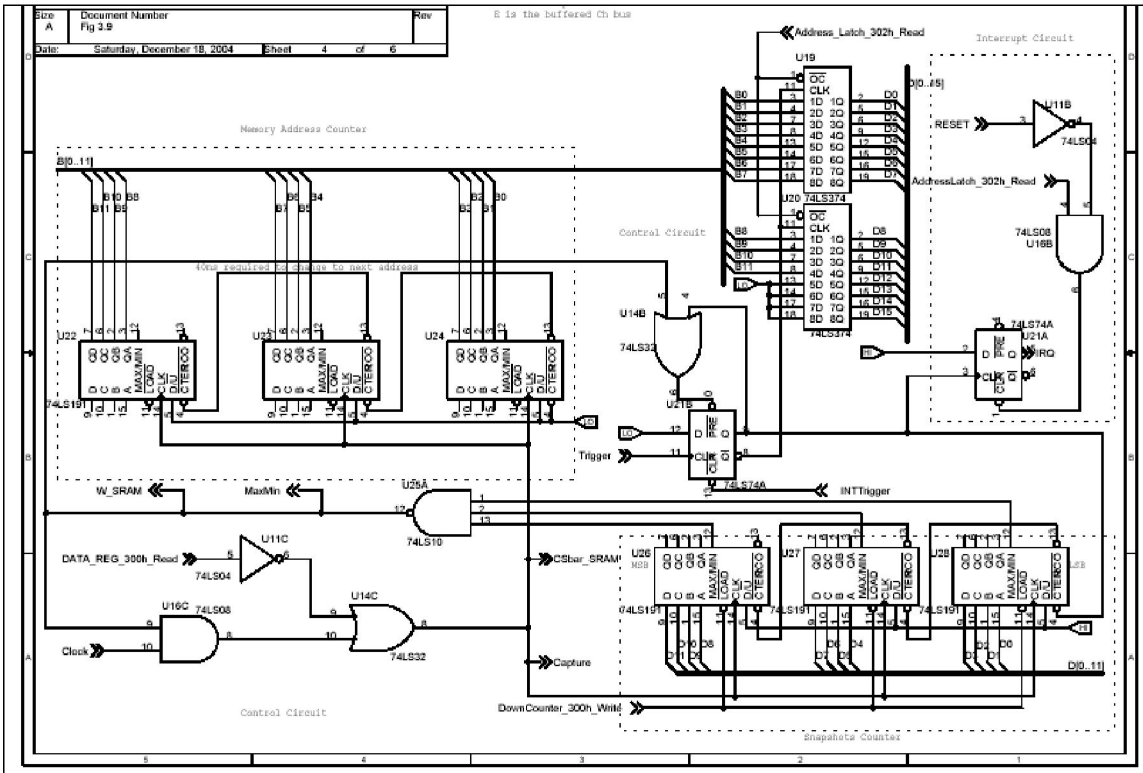
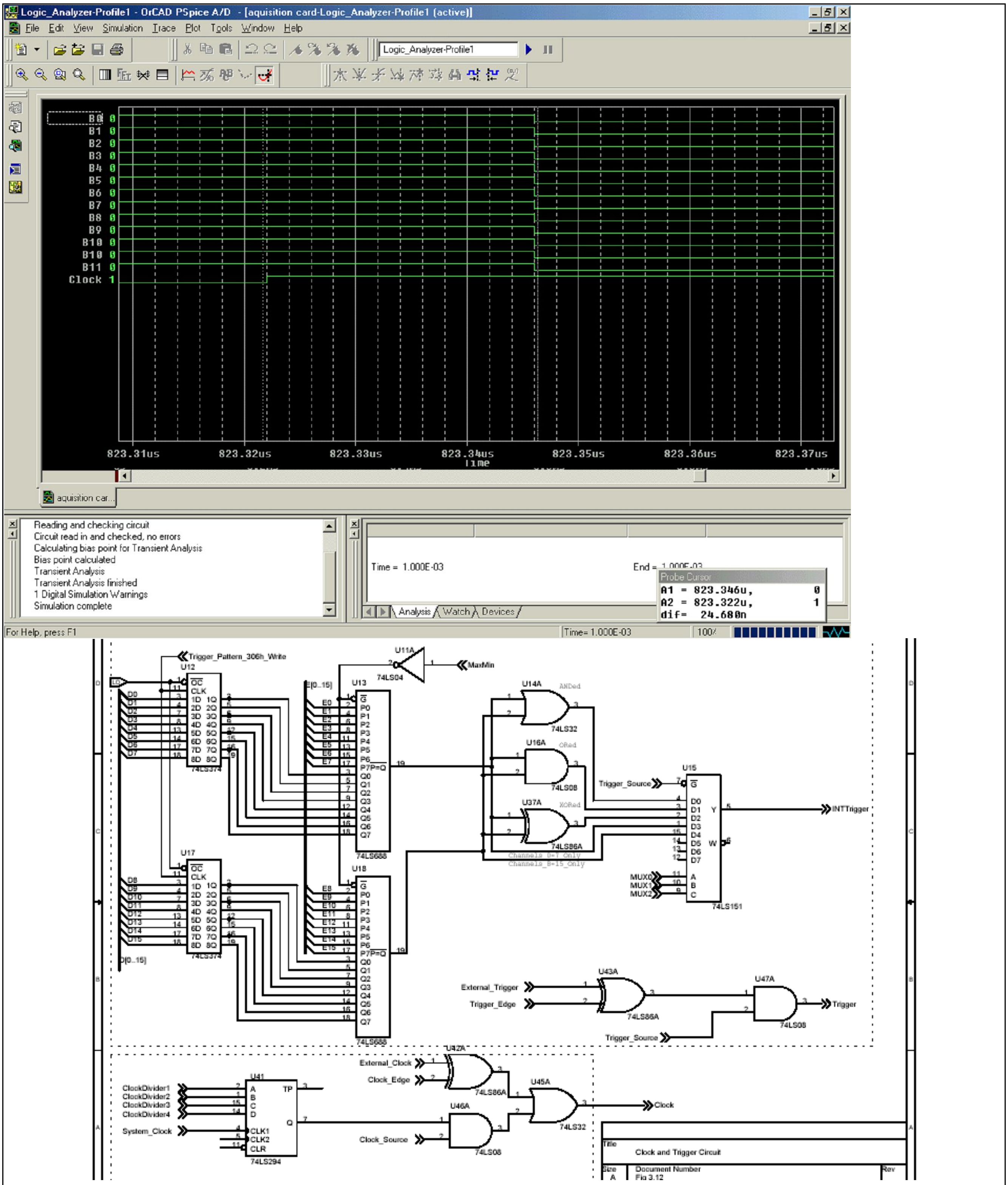


Figure (7): Main circuit.

Figure (8): Address counter timing.



```
Reading and checking circuit
Circuit read in and checked, no errors
Calculating bias point for Transient Analysis
Bias point calculated
Transient Analysis
Transient Analysis finished
1 Digital Simulation Warnings
Simulation complete
```

```
Time = 1.000E-03                End = 1.000E-03
```

```
Probe Cursor
A1 = 823.346u,      0
A2 = 823.322u,      1
dif=  24.680n
```

Figure (9): Clock and trigger circuit.

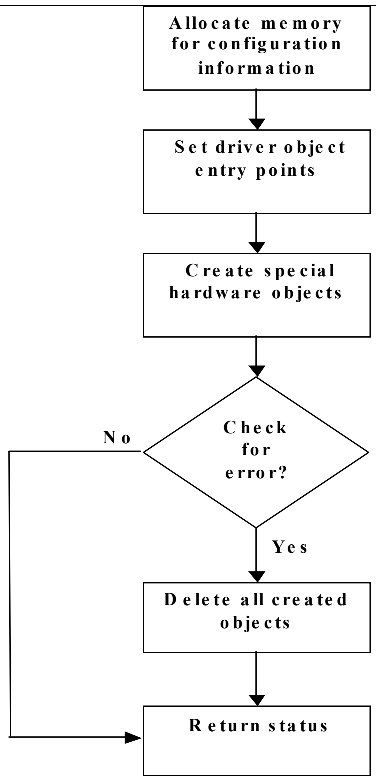

Allocate memory for configuration information
Set driver object entry points
Create special hardware objects
Check for error?
No
Yes
Delete all created objects
Return status

Figure (10): Driver Entry routine flow chart

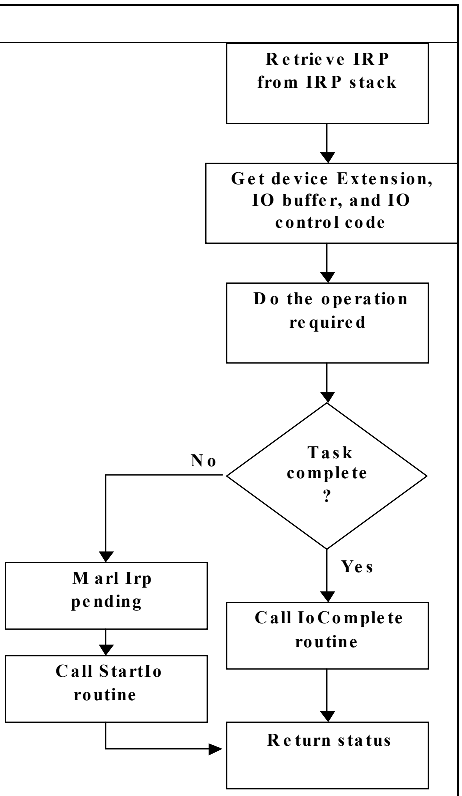

Retrieve IRP from IRP stack
Get device Extension, IO buffer, and IO control code
Do the operation required
Task complete ?
No
Yes
Marl Irp pending
Call StartIo routine
Call IoComplete routine
Return status

Figure (11): Dispatch I/O routine flow chart

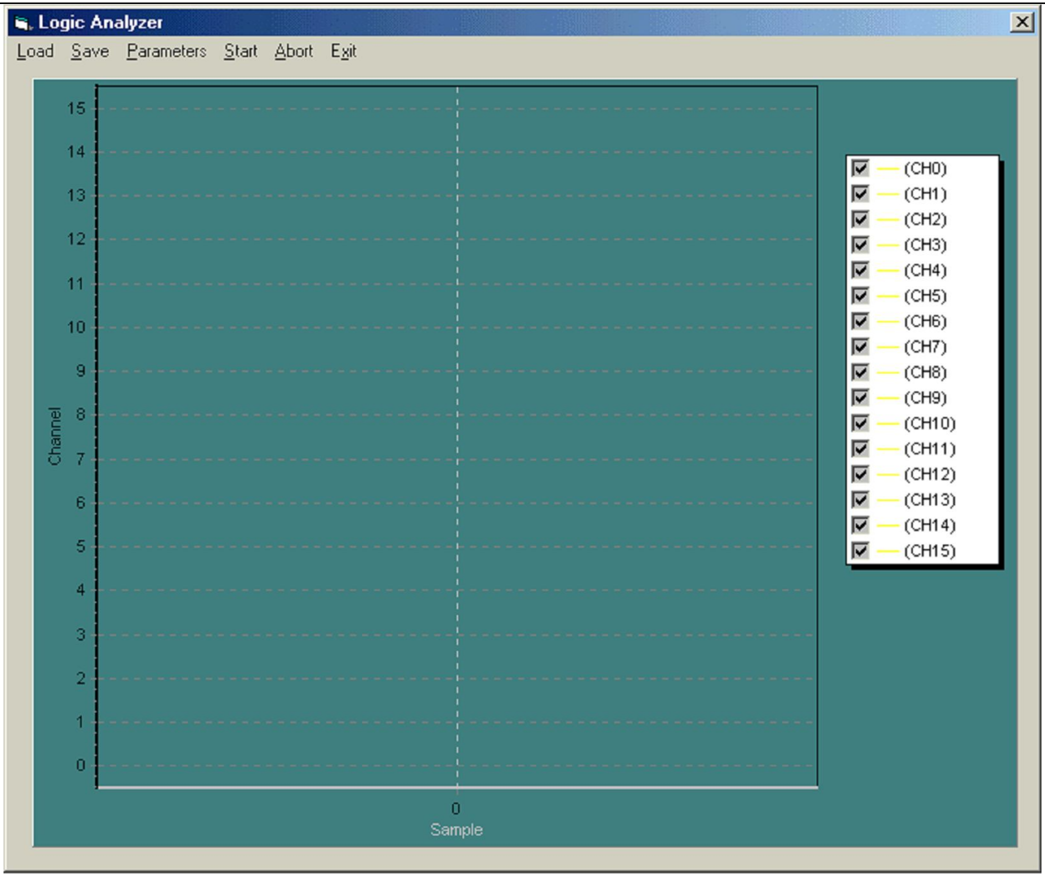
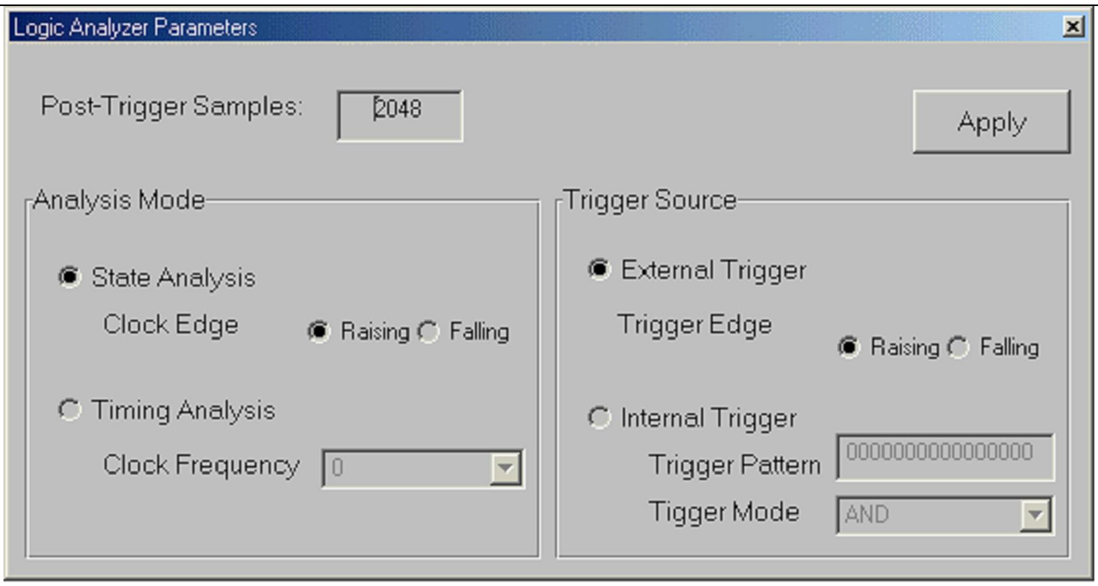


Figure (12): Graphical User Interface program



Post-Trigger Samples: 2048

Analysis Mode
● State Analysis
Clock Edge ● Raising ○ Falling
○ Timing Analysis
Clock Frequency 0

Trigger Source
● External Trigger
Trigger Edge ● Raising ○ Falling
○ Internal Trigger
Trigger Pattern 0000000000000000
Tigger Mode AND

Figure (13): Parameter setting window

Table (1) Tests type of the logic analyzer.

Test No.	Measurement Type	Pre-trigger Snapshots	Post-trigger Snapshots	Trigger Source*	Capture Clock Type
1	State	2048	2048	External	Rising Edge
2	State	2048	2048	External	Falling Edge
3	State	0	4096	Internal (channel 0-7)	Rising Edge
4	State	0	4096	Internal (ANDed)	Rising Edge
5	Timing	1000	3096	External	4 MHz clock
6	Timing	0	4096	Internal (ANDed)	4 MHz clock

Table (2) Logic analyzer channels connection.

Channel No	Signal Name
Channel 0:	AD0
Channel 1:	AD1
Channel 2:	AD2
Channel 3:	AD3
Channel 4:	AD4
Channel 5:	AD5
Channel 6:	AD6
Channel 7:	AD7
Channel 8:	ALE
Channel 9:	IO/-M
Channel 10	-RD
Channel 11:	-WR
Channel 12:	Not Connected
Channel 13:	Not Connected
Channel 14:	Not Connected
Channel 15:	Not Connected

Table (3) Test sample program.

Address	Label	Opcode & Operand	Machine Code
2000	b:	MVI C,0Fh	0E0Fh
2002	a:	DCR C	0Dh
2003		JNZ a	C20220h
2006		JMP b:	C30020h

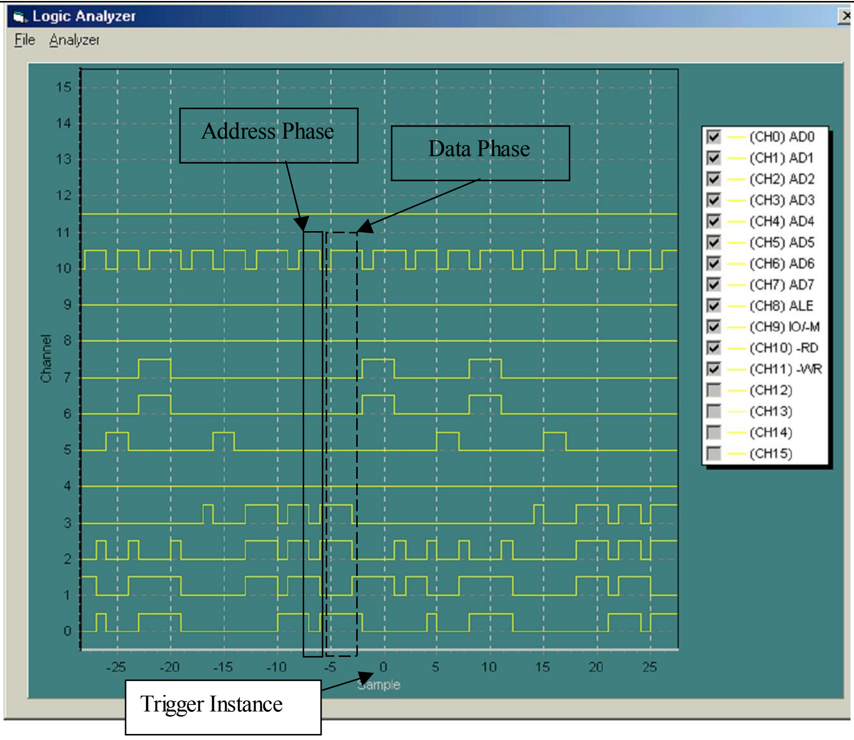


Figure (14): Zoomed results of test 1.

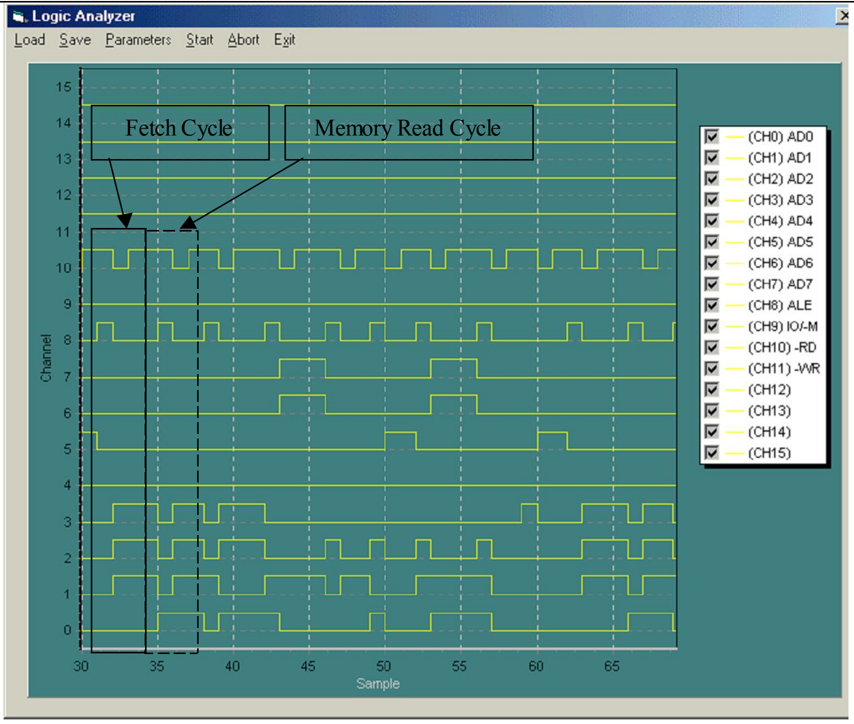


Figure (15): Results of test 2.