

MICROCONTROLLER AND DIGITAL SIGNAL PROCESSING LABORATORY

LAB MANUAL

Academic Year : 2019 - 2020
Course Code : AEC114
Regulations : IARE - R16
Class : VI Semester (EEE)

Prepared by

Ms.J.Sravana

Assistant Professor



Department of Electrical & Electronics Engineering

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal – 500 043, Hyderabad



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal – 500 043, Hyderabad

Electrical and Electronics Engineering

Vision

To produce professionally competent Electronics and Communication Engineers capable of effectively and efficiently addressing the technical challenges with social responsibility.

Mission

The mission of the Department is to provide an academic environment that will ensure high quality education, training and research by keeping the students abreast of latest developments in the field of Electronics and Communication Engineering aimed at promoting employability, leadership qualities with humanity, ethics, research aptitude and team spirit.

Quality Policy

Our policy is to nurture and build diligent and dedicated community of engineers providing a professional and unprejudiced environment, thus justifying the purpose of teaching and satisfying the stake holders.

A team of well qualified and experienced professionals ensure quality education with its practical application in all areas of the Institute.

Philosophy

The essence of learning lies in pursuing the truth that liberates one from the darkness of ignorance and Institute of Aeronautical Engineering firmly believes that education is for liberation.

Contained therein is the notion that engineering education includes all fields of science that plays a pivotal role in the development of world-wide community contributing to the progress of civilization. This institute, adhering to the above understanding, is committed to the development of science and technology in congruence with the natural environs. It lays great emphasis on intensive research and education that blends professional skills and high moral standards with a sense of individuality and humanity. We thus promote ties with local communities and encourage transnational interactions in order to be socially accountable. This accelerates the process of transfiguring the students into complete human beings making the learning process relevant to life, instilling in them a sense of courtesy and responsibility.



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

Electrical & Electronics Engineering

Program Outcomes	
PO1	Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary Settings.
PO10	Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.
Program Specific Outcomes	
PSO 1	Problem Solving: Exploit the knowledge of high voltage engineering in collaboration with power systems in innovative, dynamic and challenging environment, for the research based team work.
PSO 2	Professional Skills: Identify the scientific theories, ideas, methodologies and the new cutting edge technologies in renewable energy engineering, and use this erudition in their professional development and gain sufficient competence to solve the current and future energy problems universally.
PSO 3	Modern Tools in Electrical Engineering: Comprehend the technologies like PLC, PMC, process controllers, transducers and HMI and design, install, test, maintain power systems and industrial applications..



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal – 500 043, Hyderabad

ATTAINMENT OF PROGRAM OUTCOMES & PROGRAM SPECIFIC OUTCOMES			
S. No.	Experiment	Program Outcomes Attained	Program Specific Outcomes Attained
1	Design and develop an assembly language program using 8086 microprocessor and to show the following aspects, programming execution debugging to demonstrate the tool chain for WIN862 and hardware for 8086 microprocessor.	PO1, PO2	PSO1
2	a) Write an ALP program to perform 8 Bit arithmetic operations using 8051 b) Write an ALP program to perform 16 Bit arithmetic operations using 8051	PO1, PO2	PSO1
3	a) write an ALP program to count the number of ones in any number b) Write an ALP program to count the number of zeros in any number	PO1, PO2	PSO1
4	Write an ALP program and verify timer/counter in 8051	PO1, PO2	PSO1, PSO2
5	Write an ALP program to interface 8051 and keyboard	PO1, PO2	PSO1
6	a) Generation of linear convolution without using built in function in MATLAB b) Generation of circular convolution without using built in function in MATLAB	PO1, PO2, PO3	PSO1, PSO2
7	Compute the Discrete Fourier Transform and IDFT with and without fft and ifft in MATLAB	PO1, PO2, PO3	PSO1
8	Determination of power spectrum of a given sequence.	PO1, PO2	PSO1
9	Implementation of Decimation-in-time radix-2 FFT algorithm		
10	Implementation of Decimation-in-frequency radix-2 FFT algorithm	PO1, PO2	PSO1
11	Implementation of LP/HP IIR digital filter	PO1, PO2, PO3	PSO1
12	Implementation of LP/HP FIR digital filter	PO1, PO2	PSO1



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

Certificate

This is to Certify that it is a bonafied record of Practical work done by Sri/Kum. _____ bearing the Roll No. _____ of _____ Class _____ Branch in the _____ laboratory during the Academic year _____ under our supervision.

Head of the Department

Lecture In-Charge

External Examiner

Internal Examiner



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal – 500 043, Hyderabad

Electrical and Electronics Engineering

Course Overview:

The aim of this course is to conduct experiments on microprocessor and microcontrollers and interfacing 8051 microcontrollers to lcd, keyboard. This course is also useful for learning basic signals by using mat lab software.

Course Objective:

1. Develop assembly language program for arithmetic and logical operations using 8051.
2. Implement convolution using MATLAB.
3. Implement digital signal processing algorithms using MATLAB



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous) Dundigal
500 043, Hyderabad

Electrical and Electronics Engineering

INSTRUCTIONS TO THE STUDENTS

1. Students are required to attend all labs.
2. Students should work individually in the hardware and software laboratories.
3. Students have to bring the lab manual cum observation book, record etc along with them whenever they come for lab work.
4. Should take only the lab manual, calculator (if needed) and a pen or pencil to the work area.
5. Should learn the prelab questions. Read through the lab experiment to familiarize themselves with the components and assembly sequence.
6. Should utilize 3 hour's time properly to perform the experiment and to record the readings. Do the calculations, draw the graphs and take signature from the instructor.
7. If the experiment is not completed in the stipulated time, the pending work has to be carried out in the leisure hours or extended hours.
8. Should submit the completed record book according to the deadlines set up by the instructor.
9. For practical subjects there shall be a continuous evaluation during the semester for 25 sessional marks and 50 end examination marks.
10. Out of 25 internal marks, 15 marks shall be awarded for day-to-day work and 10 marks to be awarded by conducting an internal laboratory test.

LIST OF EXPERIMENTS	
Week-1	DESIGN A PROGRAM USING WIN862 AND 8086 MICROPROCESSOR
Design and develop an assembly language program using 8086 microprocessor and to show the following aspects, programming execution debugging to demonstrate the tool chain for WIN862 and hardware for 8086 microprocessor.	
Week-2	8 AND 16 BIT ARITHMETIC OPERATIONS
a)Write an ALP program to perform 8 Bit arithmetic operations using 8051 b)Write an ALP program to perform 16 Bit arithmetic operations using 8051	
Week-3	NUMBER OF ZEROS AND ONES IN ANY NUMBER
a)write an ALP program to count the number of ones in any number b)Write an ALP program to count the number of zeros in any number	
Week-4	TIMER / COUNTER IN 8051
Write an ALP program and verify timer/counter in 8051	
Week-5	UART OPERATION IN 8051
Write an ALP program to operate UARE in 8051.	
Week-6	INTERFACE SEVEN SEGMENT DISPLAY
Write an ALP program to interface 8051 and keyboard	
Week-7	ADC, DAC WITH 8051
a)write an ALP program to convert analog signal to digital signal using 8051 b)write an ALP program to convert digital signal to analog signal using 8051	
Week-8	CONVOLUTION
a)Generation of linear convolution without using built in function in MATLAB b)Generation of circular convolution without using built in function in MATLAB	
Week-9	DISCRETE FOURIER TRANSFORM
Compute the Discrete Fourier Transform and IDFT with and without fft and ifft in MATLAB	
Week-10	POWER SPECTRUM
Determination of power spectrum of a given sequence.	
Week-11	DIT - FAST FOURIER TRANSFORM
Implementation of Decimation-in-time radix-2 FFT algorithm	
Week-12	DIF - FAST FOURIER TRANSFORM
Implementation of Decimation-in-frequency radix-2 FFT algorithm	
Week-13	IIR FILTER
Implementation of LP/HP IIR digital filter	
Week-14	FIR FILTER
Implementation of LP/HP FIR digital filter	

Text Books:

1. Kenneth.J.Ayala. The 8051 microcontroller, 3rd Edition, Cengage learning, 2010.
2. D V Hall, “Microprocessors and Interfacing”, Tata McGraw-Hill Education, 3rd Edition 2013.
3. A K ray and K M Bhurchandani, “Advanced microprocessors and peripherals”, Tata McGraw-Hill Education, 2nd Edition 2006

Reference Books:

- 1.Fundamentals of Digital signal processing - LoneyLudeman, John wiley, 2009.
- 2.Digital signal processing: fundamentals and applications - Li Tan Elsevier, 2008.

PROCEDURE AND STEPS FOR USING WIN862

INTRODUCTION:

Features of the ESA -86/88 Microprocessor Trainer

- 8086 CPU operating at 8 MHz MAX mode.
- Provision for on-board 8087 (NDP) coprocessor.
- Provision for 256 KB of EPROM & 128 KB of RAM onboard
- Battery backup facility for RAM.
- 48 programmable I/O lines using two 8255's
- Timer1 & Timer2 signals are brought out to header pins
- Priority Interrupt Controller (PIC) for eight input using 8259A
- In standalone mode using on board keypad or with PC compatible system through its RS-232 interface
- Display is 8 seven segment LED
- Designed & engineered to integrate user's application specific interface conveniently at a minimum cost.
- Powerful & user-friendly keyboard / serial monitor, support in development of application programs.
- Software support for development of programs on Computer, the RS-232C interface cable connecting to computer from the kit facilitates transfer of files between the trainer kit & computer for development & debugging purposes.
- High quality reliable PCB with solder mask on both sides & clear legend prints with maximum details provided for the user.

SPECIFICATIONS:

CPU: Intel 8086 operating at 8 MHz in MAX mode.

MEMORY: Total 1MB of memory is in the Kit provided.

EPROM: 4 JEDEC compatible sockets for EPROM

RAM: 4 JEDEC compatible sockets for RAM

PARALLEL I/O: 48 I/O lines using two 8255

SERIAL I/O: One RS-232C compatible interface Using UART 8251A

TIMER: Three 16 bit counter / timers 8253A Counter 1 is used for serial I/O Baud rate generation.



8086 Trainer Kit

PIC: Programmable Interrupt controller using 8253A provides interrupts Vectors for 8 jumpers“ selectable Internal /External sources.

KEYBOARD / DISPLAY:

Keyboard: keyboard on to the trainer.

Display:8 seven segment displays

INTERRUPTS:

NIM: Provision for connecting NMI to a key switch

INTR: Programmable Interrupt controller using 8259A provides Interrupt vectors for 8 jumpers selectable Internal/ External Sources.

INTERFACE BUS SIGNALS:

CPU BUS: All address, data & control lines are TTL compatible & are terminated in berg strip header.

PARALLEL I/O: All signals are TTL compatible & Terminated in berg strip header For PPI expansion.

SERIAL I/O: Serial port signals are terminated in Standard 9-pin „D type connector.

MONITOR SOFTWARE:

128KB of serial / Keyboard monitor with Powerful commands to enter verify and Debug user programs, including onboard Assemble and disassemble commands.

COMPUTER INTERFACE:

This can be interfaced to host computer System through the main serial port, alsoFacilitates uploading, downloading of Intel Hex files between computer and the trainer.

I/O decoding:

IC U30 is used for on card I/O decoding. The following table gives the list of on card I/O devices and their address map.

I/O device	I/O address	I/O register	usage
8255 I (U14)	FFC0	PORT A	AVAILABLE TO USER
	FFC2	PORT B	
	FFC4	PORT C	
	FFC6	CONTROL PORT	
8255 II (U15)	FFC1	PORT A	AVAILABLE TO USER
	FFC3	PORT B	
	FFC5	PORT C	
	FFC7	CONTROL PORT	
8253 A(U28)	FFC9	TIMER 0	AVAILABLE TO USER
	FFCB	TIMER 1	USED FOR BAUD RATE
	FFCD	TIMER 2	AVAILABLE TO USER
	FFCF	CONTROL	AVAILABLE TO USER
8251A (U13)	FFD0	DATA COMMAND PORT STATUS	
	FFD2		
INPUT PORT TO DIP SWITCH (SW1)		USED AS I/P PORT TO READ SW1 AND CONFIGURE 86ME	
8259A (U12)	FFD8 TO FFDE	PRIORITY INTERRUPT CONTROLLER	

POWER REQUIREMENTS:

+5V DC with 1300 mA current rating (Max).

OPERATING CONFIGURATION:

Two different modes of operation trainer are possible. They are

- (i) Serial operation
- (ii) Keypad operation

The first configuration requires a computer system with an RS-232C port, can be used as the controlling device. When a computer system is interfaced to trainer, the driver program must be resident in the computer system.

The second mode of operation is achieved through Onboard KEYBOARD / DISPLAY. In this mode, the trainer kit interacts with the user through a computer keyboard and 16x2 LCD Display. This configuration eliminates the need for a computer and offers a convenient way for using the trainer as a stand – alone system.

EXECUTION PROCEDURE FOR 8086 (for registers):

- i) Writing a alp PROGRAM into processor:
 - Switch On Power Supply
 - Check if DIP switches board is in serial or keyboard mode (Serial mode = 1 on, Board mode = 4 On)
 - Press Reset
 - Press „EB“(Examine Byte)
 - Enter Starting Memory location (Ex: 2000)
 - Press next button, Enter OP-Code value
 - Then press next button Enter 2nd memory location and op code
 - .
 - .
 - .
 - Enter up to nth values

Execution:

- Press Exec. Button
- Press Go enter starting memory location
- Press Exec.
- Press ER (Examine Register)
- Press AX (Now see the result in Ax)

EXECUTION PROCEDURE FOR 8086 (for memory locations):

- ii) Writing a alp PROGRAM into processor:

Switch On Power Supply

Check if DIP switches board is in serial or keyboard mode (Serial mode = 1 on, Board mode

= 4

On)

Press Reset

Press „EB“ (Examine Byte)

Enter Starting Memory location (Ex: 2000)

Press next button, Enter OP-Code value

Then press next button Enter 2nd memory location and op code

.

.

.

Enter up to nth values

Execution:

Press Exec. Button

Press Go enter starting memory location

Press Exec.

Press EB give input memory location and input values

Press Exec.

Press Go Give starting memory location

Press Exec.

Press Go Now observe the results in memory location

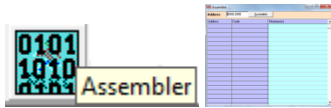
WIN862 Software procedure:

Registers:

Step 1: Open Win862 icon on desktop (see Fig.1) and opened Window see fig. 2



Step 2: Click on Assembler and give starting address (Like 0000:4000) then press Enter button.



Step 3: Then write 1st Instruction then press enter button.

Step 4: Then write 2nd Instruction then press enter button.

Step 5: Then write up to nth Instruction then press enter button and close the Assembler window.

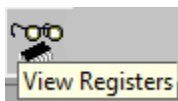
Step 6: Now click on Dis Assembler and give starting address (Like 0000:4000) then press enter button.



Step 7: Click on Set PC then give starting address then press Enter button.

Step 8: Click on Run (check whether program is executed or not)

Step 9: Click on view registers and observe the results in registers.



Memory locations:

Step 1: Open Win862 icon on desktop.

Step 2: Click on Assembler and give starting address (Like 0000:4000) then press Enter button.

Step 3: Then write 1st Instruction then press enter button.

Step 4: Then write 2nd Instruction then press enter button.

Step 5: Then write up to nth Instruction then press enter button and close the Assembler window.

Step 6: Now click on Dis Assembler and give starting address (Like 0000:4000) then press enter button.

Step 7: Click on Set PC then give starting address then press Enter button.

Step 8: Click on Run (check whether program is executed or not)

Step 9: Click on view memory

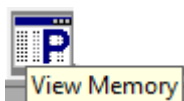
Step 10: Now enter input address

Step 11: Click on Modify and Give desired input values

Step 12: Click on Set PC. Enter initial address and press Dis-Assembler

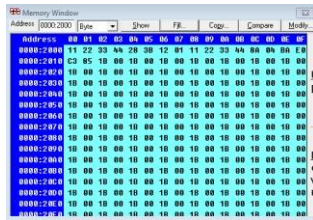
Step 13: Click on Run (check whether program is executed or not)

Step 14: Now observe the result in view memory.



Step 15: Click on view memory and enter destination address then press enter button

Step 16: Now observe the result.



INTRODUCTION OF ALS SDA 8051-MEL:



The Intel's family of 8bit single chip microcontroller has become very popular because of their unique and powerful instruction set, architecture and over all philosophy. The 8051 family has three members: 8031,8051 and 8751.the 8031 have no on-chip program memory execution is from external program memory. The 8051 has 4k bytes of factory masked ROM and has the 8751 has 4k bytes of EPROM.

The SDA 51-MEL is a System Design Aid for learning the operation of these Microcontroller devices. It uses 8031/51 as the controller. It is designed to assist students and engineers in learning about the architecture and programming of 8031/51 and designing around this Microcontroller.

The address and data bus controllers separate the 8051 microcontroller multiplexed address/data bus, creating a 16 bit address bus and 8bit data bus.

The monitor program for the SDA 51-MEL is contained in 32kbytes EPROM. The monitor interacts with the user through a CRT terminal host computer system connected through serial I/O interface or through the PC Keyboard (AT) and 16X2 LCD display.

SPECIFICATIONS

CPU: 8051 operating at 11.0592MHZ

MEMORY: EPROM1-one JEDEC compatible 28 pin socket to provide up to 32Kbytememory using 27256 with monitor software.

EPROM2-optional-canbe used as program memory, if ram is configured as data only.

RAM1-one JEDEC compatible 28 pin socket to provide up to 32Kbytes of Data memory using 62256.

RAM2-one JEDEC compatible 28 pin socket to provide up to 32Kbytes Program/data or data memory.

I/O PARALLEL: 48 I/O lines using two 8255, terminated in two 26 headers.

I/O SERAL: One RS232 compatible interface, using one chip UART lines. The lines are terminated in a 9-pin D-type female connector.onchip UART lines are also terminated in a 10 pin FRC connector.

TIMER: Three 16 bit counter/timer using 8253programmable timers terminated in a 20pin berg stick.

KEYBOARD: EXTERNAL PC –AT keyboard

DISPLAY: Alpha numeric LCD module (2linex 16 CHARS)

BUS SIGNALS: All address data and control signals are terminated in a 50 pin header Connector for user expansion. Controller specific lines like port lines T0,T1, INT1 etc are terminated in this connector.

MONITOR SOFTWARE: 32Kbytes of user of user friendly monitor software (27256) that allows Program enter, verification, debugging and execution from the system keyboard or a CRT Terminal or a PC functioning as a terminal. File uploading/downloading option is in serial mode

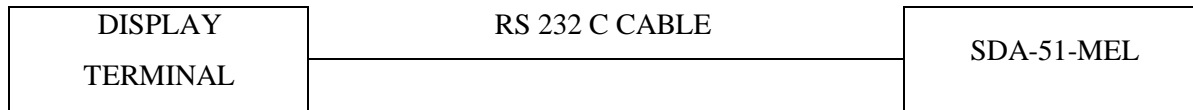
THE EXTERNAL PC: AT keyboard allows users to directly assemble /disassemble mnemonics/instructions for 8051 using the alphanumeric LCD display

OPERATING CONFIGURATION

Two different modes of Operation SDA -51MEL are possible. They are

- **serial operation**

This configuration requires an RS232 compatible terminal as the display and command entry device.



A computer system is interfaced to SDA51-MEL, a driver program must be resident in the computer system. Driver program (b30drv for DOS,TALK setup for windows) for interfacing SDA-51 MEL to a PC has been developed by ALS and is available to the user as an optional accessory.

Keyboard Operation

This mode of operation is achieved through on board KEYBOARD/DISPLAY. In this mode,SDA-51 MEL interacts with the user through an PC/AT Keyboard and a 16x2 alphanumeric LCD display. This eliminates the need for a terminal / host computer and offers a convenient way for using the SDA-51 MEL as a “STAND –ALONE” system.

SERIAL MODE:

SERIAL COMMUNICATION AND SERIAL UTILITIES

OPERATING INSTRUCTIONS

To invoke this mode press < RES> and then the < ESLR> key on the KEYBOARD to transfer control to the CRT terminal/HOST computer the prompt “SERIAL MODE” will be appears on LCD DISPLAY > ALS 8051/31 MONITER V1.0 is displayed on the terminal to indicate that the system interrogation mode and ready to accept the command. All command that be entered through interrogation modes.

SYSTEM MONITER

The SDA-51-MEL operation is controlled by monitor program stored in 32kbytes of EPROM (U5, 27256), located at SDA-51 MEL memory map (0000-7FFF).The system executes the monitor program when ever power is turn ON or when RESET is pressed.

In serial mode, the monitor program allows the user to perform following operations,

- Communicate with the SDA-51-MEL through the CRT terminal/HOST computer, using the on board serial I/O interface.
- Executes user programmers in real time or single step.

- SET break points on program,
- Examine and modify memory locations, registers and bits in SDA-51-MEL on board program/data memory and in the 8051's on chip data and register memories.
- Upload and download programmers from host computer system like PC/XT/AT(in INTEL HEX FORMAT only)

COMMANDS AVAILABLE

HELP

Syntax: H

Gives the details of the commands used in serial mode of communication.

DISPLAY COMMAND

The command is used to display the contents of register, bit memory, internal memory, program memory and external data memory

Syntax: D

On entering this command at the monitor command prompt, the following options are displayed.

DISPLAY(R, B, M, P, D)

The options are,

R for Registers,

B for bit memory

M for Internal memory

P for program memory

D for data memory

Press „enter“ to terminate the command.

EDIT COMMANDS

This command is used to edit the contents of register, bit memory, internal memory, program memory and the external data memory.

Syntax: E

On entering the command letter at the monitor command the following options are displayed.

EDIT (R, B, M, P and D).

The options are,

R for registers

P for program memory

B for bit memory

M for internal memory

D for data memory

During editing, the following keys can be used.

P to display the previous location **N** or space bar to display the next location **CR** to update and display the same location.

All other keys except 0 to 9 and A to F can be used to abort the command.

PROGRAM EXECUTION COMMANDS:

The following commands are used to control the execution of user programs. The B and C commands set and clear breakpoint address. The GO and Step commands cause the system to enter execution mode from interrogation mode.

G command:

The **G** command initiates program execution at real time (12MHZ crystal, 1micro-second cycle). The real-time execution mode allows the user to run the user code stored in program memory. Execution begins when the user enters a go command in interrogation mode. Real-time execution can be controlled by breakpoints set by the user. If program halts after executing the instruction that contained the breakpoints address, then it returns to the interrogation mode .if the breakpoints are not used, the program runs until the user terminates execution with a call to the address 0003H.

The different formants of this command and their functions are described below.

8051>G

Enter start address: 8000

This command begins real time execution of the user program beginning with the instruction currently addressed by the program counter. During program execution, the following message is displayed on the screen:

PROGRAM EXECUTION:

Execution continues until one of the following occurs:

A break point is encountered (applies only when breakpoints are enabled)

The program attempts to execute across location 0003H.this location is reserved for system operation.

After execution if break point were not specified, then all the register contents will be displayed and the monitor comes back to interrogation mode with the prompt „8051>“ meaning that the it is ready to a accept the next command

Note:

- The system uses the current program counter address as the start address.
- If program breakpoint or data breakpoint have been enabled then the program will be executed the command is terminated without execution of the program.

SINGLE STEL COMMAND:

This command executes one instruction at the address in the program counter

8051>S

8051>enter star address=8000<CR>

After each instruction, the system displays the values of the updated program counter, accumulator, data, pointer register, and stack pointer. To terminate this command press ESC or SPACE BAR. The actual format & the output of each of the instruction is given in the section serial communication demo

BREAK COMMAND

SET BREAK COMMAND:

SYNTAX 8051

Set breakpoint: up to eight breaks can be set in the user program. After giving the command „B“ at the prompted with the break number, enter the break no between 1 to 8.press <CR> after the break no. And enter the break address and press <CR>to go to conform the address and press another<CR>to go to the next break address selection or <SP><SP>to terminate the command.

CLEAR BREAKPOINTS

SYNTAX :> C

This command prompts the user for the break no,which has to be cleared. To clear all break points, enter the break number has to be cleared.

FILE UPLOAD FROM SDA-MEL TO PC

This option allows the user to save any program in memory as file in Intel hex format. On entering the command „F10“ and select option 4 on following this, the driver program prompts for the name of the file in which the data is to be stored and enter the START & END address and press,<CR>.the program assumes a default extension of HEX for the file. This system then receives the data and stores it in the specified file and on completion the main menu will be displayed.

Ex: F10

Select option 4

Enter the file name in which the data is to be stored.

Enter START address = 8000 <CR>

ENTER END address = 805F <CR>

FILE DOWNLOAD FROM PC TO SDA-EL-MEL

This option allows the user to transfer an Intel hex file on a floppy diskette to program/data memory. On processing „:“ key, the following message is displayed.

Go to the main menu by pressing F10 and select option 3

On following this, the driver program prompts for the name of the file to be downloading. Enter the file name and press <CR>. While the transfer operation in progress, the system displays the number record be transferred.

At the end of the transfer the main menu is displayed. Go to terminal mode press <CR>, the following message

File received O.K. will be displayed

Ex:“:“

Go to the main menu by pressing F10 and select option 3

KEY BOARD MODE OF OPERATION

At the power on the monitor automatically goes into keyboard mode, at power on the sign on message SDA 51/31/-STA<E> HELP appears on the LCD display.

THE FUNCTION OF SOME SPECIAL KEYS ON THE PC/AT KEYBOARD ARE LISTED BELOW

KEY LABEL	DESCRIPTION
RESET	Transfers control to the monitor at location 0000H
NXT	The monitor interrupts this key as a delimiter. Different commands are explained later .
ENTER	The monitor command terminator
BMOVE	Selects the monitor block move command
GO	Selects the monitor go command (program execution)
PREV	A monitor delimiter key, and in the next coming section its usage's are explained

STEP	Selects the monitor single step function
EREG	Selects the monitor examine / modify cpu register function
EDM	In combination with substitute memory command this key allows the using to examine and modify external data memory
IDM	In combination with substitute memory command this key allows the using to examine and modify internal data memory
IBM	In combination with substitute memory command this key allows the using to examine and modify internal bit memory
EPM	In combination with substitute memory command this key allows the using to examine external program memory
EPGM	Used to program EPROM's using EPROM programmer I/F(NIFC 03)
EPRD	Used to read the EPROM contents using EPROM programmer I/F(NIFC 03)
ESRL	Key to invoke serial mode
ASM	Key to invoke assembler mode
DSM	Key to invoke di assembler mode
BS	Provides back facility in assembler mode

SUBSTITUTE MEMORY COMMAND

This command is used to examine/modify the memory functions. This command will support examine/modification of following memories.

- ❖ External data memory (EDM)
- ❖ External program memory (EPM)
- ❖ internal data memory (IDM)
- ❖ internal bit memory (IBM)

This command is invoked using „SMEM“ key in the ASCII key board the message “SUBSTITUTE MEMORY” appears on the display.

Then user can select any one of above mention four memories, and enter the location address to be Examine/modify and press <NXT> to display the data present in that memory location, now user can modify that data byte if required then again he has to press <NXT>, now PC is incremented to show the contents of the next memory location. If the user wants to see the content of previous location i.e. if 9005H is the current PC content & he wants to see the 9004 location content then he has to press <PREV>key.

EXTERNAL DATA MEMORY

<SMEM><EDM><address of memory location><NXT><new byte if required><NXT>.....<ENTER>.

This command is used to enter the data in data memory (0300H to 1FFFH, 4000H to 7FFFH) or data/code into data/program memory (8000H to FFFFH).

INTERNAL DATA MEMORY

<SMEM><IDM><Address of the memory location><NXT><new byte if required><NXT>..... <ENTER>

Internal data memory ranges from 00H to 7FH(128bytes)

INTERNAL BIT MEMORY

<SMEM><IBM><Address of the memory location><NXT><new byte if required><NXT>..... <ENTER>

Internal bit memory ranges from 00 to 7F(128bits) values entered must be 1 or 0 only.

EXTERNAL PROGRAM MEMORY

<SMEM><EPM><Address of the memory location><NXT><NXT>...<ENTER>

If the user attempts to edit data in this region an ERROR message will be displayed.

EXAMINE/MODIFY CPU REGISTERS COMMAND

The examine/modify register command allows the user to examine/modify the contents of CPU registers. This command is invoked using EREG key in the ASCII keyboard, the message “which register?” appears on the first line of LCD display then the user can select the CPU register which he wants to examine/modify through a key designator (for the key designators see the table given below) then if <NXT> pressed the register name in the registers sequence and its content will be displayed, the registers display sequence if A,B,R0,R1,R2,R3,R4,R5,R6,R7,PCL,PCH,PSW,SP,DPH,DPL.

DESIGNATOR(KEY)	CPU REGISTERS	DESIGNATOR(KEY)	CPU REGISTERS
0	RO	8	PCL
1	R1	9	PCH
2	R2	A	A
3	R3	B	B
4	R4	C	SP
5	R5	D	DPH
6	R6	E	DPL
7	R7	F	PSW

EXECUTE USER PROGRAM COMMAND

The execute user program command allows user to execute a program that he has entered/downloaded. To invoke this execute user program command press <GO> now the current PC and its data are displayed on the LCD display and then the command is completed when the user press<ENTER>the message “PROGRAM EXECUTED” will be displayed on the LCD display.

SYNTAX: Go<Program starting address><ENTER>

EX: To execute a program which is having the starting address at 8000H<GO>8000<ENTER>

SINGLE STEP COMMAND

The single step command allows the user to „instruction step“ through his program, this command is invoked through <STEP> key when the user press<STEP> the current PC content and data of that location are displayed on the LCD module. The user can now change the address, if required and then press <ENTER >,the instruction at that address is executed and its contents are displayed, now by pressing <NXT>key the display updates to next logical address and its contents. To examine register or memory contents at this stages press<ENTER>then <EREG>/<SMEM> or any command provided to user in keyboard mode and again to enter single step press <ENTER>and to continue the stepping process press<NXT><NXT>....

In this single step mode, we use INT0 with its priority bit set. A such the other interrupts are not functional.

SYNTAX:

<STEP><Starting address of user program><ENTER><NXT><NXT>.....

EX:To single step a program with starting address 9000H,and in the third step exam register command has to be invoked to see the content of registers A,B,R0, then again come back for single stepping.

<STEP>8000<ENTER><NXT><NXT><NXT><ENTER>

<EREG><A><NXT><NXT><NXT><ENTER><STEP><ENTER><NXT>

<NXT>.....

TALK software Procedure:

First identify Location of TALK software. If it is in D drive then choose run prompt and select CMD then follow below procedure.

D:\>

ENTER

D:\>cd comm_pack86

ENTER

D:\cd comm_pack86 >cd comm_pack86

Enter

D:\cd comm_pack86 >cd comm_pack86>cd x8086

Enter

D:\cd comm_pack86 >cd comm_pack86>cd x8086>edit file name

Enter

Enter the program

Go to file and save & go to file exit

Press x8086

Enter

Then go to options disconnected and connected, press reset button in kit

Display -als-86 monitor

Go file selected download Intel hex. File<comm._pack86>,<openx8086>,<filename>open

Enter

Display #

Next selected in kit 1&7 pins keeps ON and press reset button in kit

Selected in G

Give the address and press enter

EXPERIMENT -1

DESIGN A PROGRAM USING WIN862 AND 8086 MICROPROCESSOR

AIM: -

To write an assembly language program for Addition of two 16-bit numbers.

COMPONENTS & EQUIPMENT REQUIRED: -

S.No	Device	Range / Rating	Quantity (in No's)
1	8086 microprocessor kit/Win862 with PC		1
2	Keyboard		1
3	RPS	+5v	1

PROGRAM:

```
MOV AX, 4343
MOV BX, 1111
ADD AX, BX
INT 03
```

Hardware:

MEMORY LOCATION	OP-CODE	MNEMONIC OPERAND	COMMENTS
		MOV AX,4343 MOV BX,1111 ADD AX,BX INT 3	

Observation Table

Input		Output	
Register	Data	Register	Data
AX	4343	AX	
BX	1111		

II)MULTIBYTE ADDTION

AIM: - To write an assembly language PROGRAM for MULTIBYTE ADDITION

Components & equipment required: -

S.No	Device	Range / Rating	Quantity (in No's)
1	8086 microprocessor kit/Win862 with PC		1
2	Keyboard		1
3	RPS	+5v	1

PROGRAM:

```

                                MOV AX,0000
                                MOV SI,2000
                                MOV DI,3000
                                MOV BX,2008
                                MOV CL,04
UP :                             MOV AL,[SI]
                                ADD AL,[BX]
                                MOV [DI],AL
                                INC SI
                                INC BX
                                INC DI
                                DEC CL
                                JNZ UP
                                INT 03
```

Hardware:

MEMORY LOCATION	OP-CODE	MNEMONIC OPERAND	COMMENTS
		MOV AX,0000	
		MOV SI,2000	
		MOV DI, 3000	
		MOV BX, 2008	
		MOV CL, 04	
		MOV AL, [SI]	
		ADD AL, [BX]	
		MOV [DI], AL	
		INC SI	
		INC BX	
		INC DI	
		DEC CL	

		JNZ UP INT 3	
--	--	-----------------	--

Observation Table:

Input				Output	
MEMORY LOCATION	Data	MEMORY LOCATION	Data	MEMORY LOCATION	Data
2000		2008		3000	
2001		2009		3001	
2002		200A		3002	
2003		200B		3003	
2004					
2005					
2006					
2007					

SUBTRACTION:

i) 16 bit subtraction

AIM: -

To write an assembly language PROGRAM for subtraction of two 16-bit numbers.

COMPONENTS & EQUIPMENT REQUIRED: -

S.No	Device	Range / Rating	Quantity (in No's)
1	8086 microprocessor kit/Win862 with PC		1
2	Keyboard		1
3	RPS	+5v	1

PROGRAM:

```
MOV AX, 4343
MOV BX, 1111
SUB AX, BX
```

INT 03

HARDWARE:

MEMORY LOCATION	OP-CODE	MNEMONIC OPERAND	COMMENTS
		MOV AX,4343 MOV BX,1111 SUB AX,BX INT 03	

OBSERVATION TABLE:

Input		Output	
Register	Data	Register	Data
AX	4343	AX	3232
BX	1111		

i) MULTIBYTE SUBTRACTION

AIM: - PROGRAM to perform multi byte subtraction.

COMPONENTS & EQUIPMENT REQUIRED: -

S.No	Device	Range / Rating	Quantity (in No's)
1	8086 microprocessor kit/Win862 with PC		1
2	Keyboard		1
3	RPS	+5v	1

PROGRAM:

```

MOV AX,0000
MOV SI,2000
MOV DI,3000
MOV BX,2008
MOV CL,04
UP : MOV AL,[SI]
      SUB AL,[BX]
      MOV [DI],AL
      INC SI
      INC BX
      INC DI
      DEC CL
      JNZ UP
      INT03
    
```

HARDWARE

MEMORY LOCATION	OP-CODE	LABEL	MNEMONIC OPERAND	COMMENTS
		UP	MOV AX,0000 MOV SI, 2000 MOV DI, 3000 MOV BX, 2008 MOV CL, 04 MOV AL, [SI] SUB AL, [BX] MOV [DI], AL INC SI INC BX INC DI DEC CL JNZ UP INT 03	

OBSERVATION TABLE:

Input				Output	
MEMORY LOCATION	Data	MEMORY LOCATION	Data	MEMORY LOCATION	Data
2000		2008		3000	
2001		2009		3001	
2002		200A		3002	
2003		200B		3003	
2004					
2005					
2006					
2007					

C) MULTIPLICATION

i) 16 bit multiplication

AIM: -

To write an assembly language PROGRAM for multiplication of two 16-bit numbers.

COMPONENTS & EQUIPMENT REQUIRED: -

S.No	Device	Range / Rating	Quantity (in No's)
1	8086 microprocessor kit/Win862with PC		1
2	Keyboard		1
3	RPS	+5v	1

PROGRAM:

i) **Software**

```
MOV AX, 4343
MOV BX, 1111
MUL BX
INT 03
```

Hardware

MEMORY LOCATION	OP-CODE	LABEL	MNEMONIC OPERAND	COMMENTS
			MOV AX,4343 MOV BX,1111 MUL BX INT 3	

Observation Table

Input		Output	
Register	Data	Register	Data
AX	4343	AX	EA73
BX	1111	DX	047B

ii) **16 bit multiplication (signed numbers)**

AIM: -

To write an assembly language PROGRAM for multiplication of two 16-bit signed numbers.

COMPONENTS & EQUIPMENT REQUIRED: -

S.No	Device	Range / Rating	Quantity (in No's)
1	8086 microprocessor kit/Win862with PC		1
2	Keyboard		1
3	RPS	+5v	1

PROGRAM

```

MOV SI,2000
MOV DI,3000
MOV AX,[SI]
ADD SI,02
MOV BX,[SI]
IMUL BX
MOV [DI],AX
ADD DI,02
MOV [DI],DX
INT 03

```

Hardware

MEMORY LOCATION	OP-CODE	LABEL	MNEMONIC OPERAND	COMMENTS
			MOV SI,2000 MOV DI,3000 MOV AX,[SI] ADD SI,02 MOV BX,[SI] IMUL BX MOV [DI],AX ADD DI,02 MOV [DI],DX INT 3	

Observation Table

Input		Output	
MEMORY LOCATION	Data	MEMORY LOCATION	Data
2000		3000	
2001		3001	
2002		3002	
2003		3003	

D) DIVISION

i) 16 bit division

AIM:-

To write an assembly language PROGRAM for multiplication of two 16-bit numbers.

COMPONENTS & EQUIPMENT REQUIRED: -

S.No	Device	Range / Rating	Quantity (in No's)
1	8086 microprocessor kit/Win862 with PC		1
2	Keyboard		1
3	RPS	+5v	1

PROGRAM

```
MOV AX, 4343  
MOV BX, 1111  
DIV BX  
INT 03
```

I) Hardware

MEMORY LOCATION	OP-CODE	LABEL	MNEMONIC OPERAND	COMMENTS
			MOV AX,4343 MOV BX,1111 DIV BX INT 3	

Observation Table

Input		Output	
Register	Data	Register	Data
AX	4343	AX	0003
BX	1111	DX	03F2

RESULT:

PRE LAB QUESTIONS:

- 1.How many bit 8086microprocessor is?
- 2.What is the sizeof data bus of8086?

3. What is the size of address bus of 8086?
4. What is the maximum memory addressing capacity of 8086?
5. Which are the basic parts of 8086?
6. Write an assembly program for multiplication and division of two 16-bit numbers?

POST LAB QUESTIONS:

1. How to move data from one register to other
2. To swapping the data what type register used
3. What are the advantages of maximum mode

EXPERIMENT NO 2

PROGRAMMING USING ARITHMETIC AND LOGICAL INSTRUCTIONS OF 8051

Aim:-

Write an ALP program to perform 8 bit arithmetical operations by using 8051.

COMPONENTS & EQUIPMENT REQUIRED: -

S.No	Device	Range / Rating	Quantity (in No's)
1	8051 trainer kit with keyboard		1
2	Talk with PC		1
3	RPS	+5v	1

PROGRAM FOR ADDITION:

i) **Software**

```

Org 9000h
MOV A,#02
MOV B,#02
ADD A,B
LCALL 03
    
```

ii) **Hardware**

MEMORY LOCATION	OPCODE	LABEL	MNEMONIC OPERAND
			MOV A,#02
			MOV B,#02
			ADD A,B
			LCALL 03

OBSERVATION TABLE:

Input		output	
REGISTER	Data	REGISTER	Data
A	02	A	04
B	02		

PROGRAM FOR SUBTRACTION:

i) **Software**

```
Org 9000h
MOV A,#02
MOV B,#02
SUBB A,B
LCALL 03
```

ii) **Hardware**

MEMORY LOCATION	OPCODE	LABEL	MNEMONIC OPERAND
8000			MOV A,#04 MOV B,#02 SUBB A,B LCALL 03

OBSERVATION TABLE

Input		output	
REGISTER	Data	REGISTER	Data
A	04	A	02
B	02		

PROGRAM FOR MULTIPLICATION:

i) **Software**

```
Org 9000h
MOV DPTR,#9000H
MOVX A,@DPTR
MOV F0,A
INC DPTR
MOVX A,@DPTR
MUL AB
LCALL 03
```

ii) *Hardware*

MEMORY LOCATION	OPCODE	LABEL	MNEMONIC OPERAND
8000			MOV DPTR,#9000 MOVX A,@DPTR MOV F0,A INC DPTR MOVX A,@DPTR MUL AB LCALL 03

Observation Table

Input		output	
MEMORY LOCATION	Data	REGISTER	Data
9000	03	A	06
9001	02		

PROGRAM FOR DIVISION:

i) **Software**

```

Org 9000h
MOV DPTR,#9000H
MOVX A,@DPTR
MOV R0,A
INC DPTR
MOVX A,@DPTR
MOV F0,A
MOV A,R0
DIV AB
INC DPTR
MOV @DPTR,A
LCALL 03
    
```

ii) Hardware

MEMORY LOCATION	OPCODE	LABE L	MNEMONIC OPERAND
			MOV DPTR,#9000 MOVX A,@DPTR MOV R0,A INC DPTR MOVX A,@DPTR MOV F0,A MOV A,R0 DIV AB INC DPTR MOV @DPTR,A LCALL 03

OBSERVATION TABLE:

Input		Output	
MEMORY LOCATION	Data	REGISTER	Data
9000	03	A	06
9001	02		

LOGICAL OPERATIONS:

AIM:-

To perform logical operations by using 8051.

COMPONENTS & EQUIPMENT REQUIRED: -

S.No	Device	Range / Rating	Quantity (in No's)
1	8051 trainer kit with keyboard		1
2	Talk with PC		1
3	RPS	+5v	1
4	RS – 232		1

PROGRAM FOR AND OPERATION:

i) Software

```
Org 9000h
MOV R0,#DATA 1
MOV A,#DATA 2
ANL A,R0
MOV R1,A
LCALL 03
```

ii) Hardware

MEMORY LOCATION	OPCODE	LABEL	MNEMONIC OPERAND
			MOV R0,#DATA 1 MOV A,#DATA 2 ANL A,R0 MOV R1,A LCALL 03

OBSERVATION TABLE:

Input		Output	
Register	Data	Register	Data
R0		R1	
A			

PROGRAM FOR OR OPERATION:

i) **Software**

```

Org 9000h
MOV R0,#DATA 1
MOV A,#DATA 2
ORL A,R0
MOV R1,A
LCALL 03

```

ii) **Hardware**

MEMORY LOCATION	OPCODE	LABEL	MEMONIC OPERAND
			MOV R0,#DATA 1 MOV A,#DATA 2 ORL A,R0 MOV R1,A LCALL 03

OBSERVATION TABLE:

Input		Output	
REGISTER	Data	REGISTER	Data
R0		R1	
A			

PROGRAM FOR XOR OPERATION:

i) **Software**

```

Org 9000h
MOV R0,#DATA 1
MOV A,#DATA 2

```

```
XRL A,R0
MOV R1,A
LCALL 03
```

ii) Hardware

MEMORY LOCATION	OPCODE	LABEL	MEMONIC OPERAND
			MOV R0,#DATA 1 MOV A,#DATA 2 XRL A,R0 MOV R1,A LCALL 03

OBSERVATION TABLE:

Input		output	
REGISTER	Data	REGISTER	Data
R0		R1	
A			

RESULT:

PRE LAB QUESTIONS:

1. What is the function of 01h of Int 21h?
2. What is the function of 02h of Int 21h?
3. What is the function of 09h of Int 21h?
4. What is the function of 0Ah of Int 21h?
5. What is the function of 4ch of Int 21h?

POST LAB QUESTIONS:

1. What do u mean by emulator
2. What is the size of flag register
3. What are ASCII codes for nos. 0 to F
4. Which no. representation system you have used

EXPERIMENT NO 3

NUMBER OF ZEROS AND ONE'S IN ANY NUMBER

AIM:-

Write an ALP program to Perform number of zeros and one's in any number.

COMPONENTS & EQUIPMENT REQUIRED: -

S.No	Device	Range / Rating	Quantity (in No's)
1	8051 trainer kit with keyboard		1
2	Talk with PC		1
3	RPS	+5v	1
4	RS – 232		1
5	FRC cables		1

PROGRAM FOR AND OPERATION:

HARDWARE:

```

Mov r1,#00
Mov r2,#00
Mov r7,#08
Mov a,#20
L1: RLC A
JC L2
Inc r1
Sjmp L3
L2:inc R2
L3:DJNZ R7,L1
Lcall 03

```

OBSERVATION TABLE:

MEMORY LOCATION	OPCODE	LABEL	MEMONIC OPERAND
9000			Mov r1,#00 Mov r2,#00 Mov r7,#08 Mov a,#20 L1: RLC A JC L2 Inc r1 Sjmp L3

			L2:inc R2 L3:DJNZ R7,L1 Lcall 03
--	--	--	--

OBSERVATION TABLE:

Input		output	
REGISTER	Data	REGISTER	Data
R0		R1	
A			

RESULT:

PRE LAB QUESTIONS:

1. What is the Upper 128bytes of 8051?
2. What is the PSW in 8051 ?Explain all.
3. What is the difference between 08Hand 01H functions of LCall?
4. How register banks 0,1,2,3 is selected in 8051?
5. Which is the defaultsegmentbase: offset pairs?

POST LAB QUESTIONS:

1. Why we indicate FF as 0FF in program?
2. What is a type of queue in 8086
3. While accepting no. from user why u need to subtract 30 from that?

EXPERIMENT NO 4

PROGRAM AND VERIFY TIMER/COUNTER IN 8051

AIM:-

Write an ALP program to Perform Timer 0 and Timer 1 in Counter Mode and Gated Mode operation.

COMPONENTS & EQUIPMENT REQUIRED: -

S.No	Device	Range / Rating	Quantity (in No's)
1	8051 trainer kit with keyboard		1
2	Talk with PC		1
3	RPS	+5v	1
4	RS – 232		1
5	FRC cables		1

THEORY:

The 8051 has two 16 bit timer/ counters. They can be used either as a timer or event count. Each 16 bit timer accessed as two separate registers as TL0, TL1 and TH0, TH1 bytes.

PROGRAM TO VERIFY TIMER '0'- COUNTER MODE:

i) Software

```
Org 8000h
MOV A,TMOD (TMOD=89)
ORL A,#05H
MOV TMOD,A
SETB TRO (TRO=8C)
LCALL 68EAH
Loop: MOV DPTR,#0194H
MOV A,TLO (TLO=8A)
MOVX @DPTR,A
INC DPTR
MOV A,THO (THO=8C)
MOVX @DPTR,A
LCALL 6748H
SJMP LOOP
```

Hardware

MEMORY LOCATION	OPCODE	LABEL	MNEMONIC OPERANDS
		LOOP:	MOV A,TMOD (TMOD=89) ORL A,#05H MOV TMOD,A SETB TRO (TRO=8C) LCALL 68EAH MOV DPTR,#019 4H MOV A,TLO (TLO=8A) MOVX @DPTR,A INC DPTR MOV A,THO (THO=8C) MOVX @DPTR,A LCALL 6748H SJMP LOOP

EXECUTION:

- 1) Short jp1 of 1&2 pins and press sw1 for manual increment
- 2) Short jp1 of 2&3 pins for auto increment

PROGRAM TO VERIFY TIMER '1'- COUNTER MODE:

i) **Software**

```

Org 8000h
MOV A,TMOD (TMOD=89)
ORL A,#50H
MOV TMOD,A
SETB TR1 (TR1=8E)
LCALL 68EAH
Loop: MOV DPTR,#0194H
MOV A,TL1 (TL1=8B)
MOVX @DPTR,A
INC DPTR
MOV A,TH1 (TH1=8D)
MOVX @DPTR,A
LCALL 6748H
SJMP LOOP

```

ii)Hardware

Memory location	OPCODE	LABEL	MNEMONIC OPERANDS
		LOOP:	MOVA, TMOD (TMOD=89) ORL A,#50H MOV TMOD,A SETB TR1 (TR1=8E) LCALL 68EAH MOV DPTR,#0194H MOV A,TL1 (TL1=8B) MOVX @DPTR,A INC DPTR MOV A,TH1 (TH1=8D) MOVX @DPTR,A LCALL 6748H SJMP LOOP

EXECUTION:

- 2) Short jp1 of 5&6 pins and press sw2 for manual increment
- 3) Short jp2 of 4&5 pins for auto increment

RESULT:

PRE LAB QUESTIONS:

1. What is the reset address of 8086?
2. What is the size of flag register in 8086? Explain all.
3. What is the difference between 08H and 01H functions of INT21H?
4. Which is faster- Reading word size data whose starting address is at even or at odd address of memory in 8086?
5. Which is the default segment base: offset pairs?

POST LAB QUESTIONS:

6. Why we indicate FF as OFF in program?
7. What is a type of queue in 8086
8. While accepting no. from user why u need to subtract 30 from that?

EXPERIMENT No 5

INTERFACING MATRIX/KEYBOARD TO 8051

AIM:-

Interface a Keyboard to 8051 microcontroller.

COMPONENTS & EQUIPMENT REQUIRED: -

S.No	Device	Range / Rating	Quantity (in No's)
1	8051 trainer kit with keyboard		1
2	Key board module		1
3	RPS	+5v	1
4	FRC cables		1
5	RS-232 cable		

THEORY:

8255 is a general purpose Programmable peripheral interface device. It can be used to interface keyboard with 8051 microcontroller. All the I/O devices require up to 3 I/O ports (Port A, Port B and Port A) which is provided by 8255. Interface circuit also will be simple

Port A is configured as an input port to receive the row-column code.

Port B is configured as an output port to display the key(s) pressed.

Port C is configured as an output port to output zeros to the rows to detect a key.

PROGRAM:

```

CNTRL EQU 2043H           ;CONTROL PORT ADDRESS OF 8255
PORTA EQU 2040H          ;PORTA ADDRESS OF 8255
PORTB EQU 2041H          ;PORTB ADDRESS OF 8255
PORTC EQU 2042H          ;PORTC ADDRESS OF 8255

```

Software

```

Org 9000h
MOV A,#90H
MOV DPTR,#CNTRL
MOVX @DPTR,A
MOV B,#20H
Blink 2: MOV DPTR,#PORTB
MOV A,#FFH
MOVX @DPTR,A
MOV DPTR,#PORTC
MOV A,#00H

```

```

MOVX @DPTR,A
MOV A,#F0H
MOVX @DPTR,A
DJNZ B,BLNK2
Back: MOV A,#FEH
MOV B,#21H
Blink1: MOV DPTR,#PORTB
MOVX @DPTR,A
MOV DPTR,#PORTC
MOV A,#00H
MOVX@DPTR,A
MOV A,#F0H
MOVX @DPTR,A
LCALL DELAY
RL A
DJNZ B,BLNK1
SJMP BACK
Delay: MOV R0,#F7H
Oloop: MOV R1,#FFH
Iloop: DJNZ R1,ILOOP
DJNZ R0,OLOOP
RET

```

ii) Hardware

Memory Location	OPCODE	LABEL	MNEMONIC OPERANDS
			MOV A,#90H
			MOV DPTR,#CNTRL
			MOVX @DPTR,A
			MOV B,#20H
		BLINK2:	MOV DPTR,#PORTB
			MOV A,#FFH
			MOVX @DPTR,A
			MOV DPTR,#PORTC
			MOV A,#00H
			MOVX @DPTR,A
			MOV A,#F0H
			MOVX @DPTR,A
			DJNZ B,BLNK2
			MOV A,#FEH
		BACK:	MOV B,#21H
			MOV DPTR,#PORTB

		<pre> BLINK1: MOVX @DPTR,A MOV DPTR,#PORTC MOV A,#00H MOVX@DPTR,A MOV A,#F0H MOVX @DPTR,A LCALL DELAY RL A DJNZ B,BLNK1 SJMP BACK MOV R0,#F7H MOV R1,#FFH DJNZ R1,ILOOP DJNZ R0,OLOOP RET OLOOP: ILOOP: </pre>
--	--	--

RESULT:

PRE LAB QUESTIONS:

5. What is the size of flag register?
6. Can you perform 32 bit operation with 8086? How?
7. Whether 8086 is compatible with Pentium processor?
8. What is 8087? How is it different from 8086?
9. While accepting no. from user why do you need to subtract 30 from that?

POST LAB QUESTIONS:

1. Compare memory interfacing and IO interfacing
2. How are the even odd addresses for 8086

EXPERIMENT No 6

LINEAR CONVOLUTION AND CIRCULAR CONVOLUTION

Aim: Generation of linear convolution without using built in function and the function convolution in MATLAB

TOOLS REQUIRED:

1. Mat lab software
2. Personal computer

Program

```
clc;
close all
clear all
x=input('Enter x:      ')
h=input('Enter h:      ')

m=length(x); n=length(h);

X=[x,zeros(1,n)];

H=[h,zeros(1,m)]; for
i=1:n+m-1 Y(i)=0;
for j=1:m if(i-
j+1>0)
Y(i)=Y(i)+X(j)*H(i-j+1);
else
end
end
end Y

stem(Y); ylabel('Y[n]');
xlabel('----- >n');

title('Convolution of Two Signals without conv function');
```

CIRCULAR CONVOLUTION

Aim: Generation of circular convolution without using built in function in MATLAB

TOOLS REQUIRED:

3. Mat lab software
4. Personal computer

Program

```
Clc;

close all
clear all
x=input('Enter x:          ')
h=input('Enter h:          ')
m=length(x); n=length(h);
X=[x,zeros(1,n)];
H=[h,zeros(1,m)]; for
i=1:n+m-1 Y(i)=0;
for j=1:m if(i-
j+1>0)
Y(i)=Y(i)+X(j)*H(i-j+1);
else
end
end
end
end
Y
stem(Y); ylabel('Y[n]');
xlabel('----- >n');
title('Convolution of Two Signals without conv function');
```

OUTPUT AND WAVEFORMS:

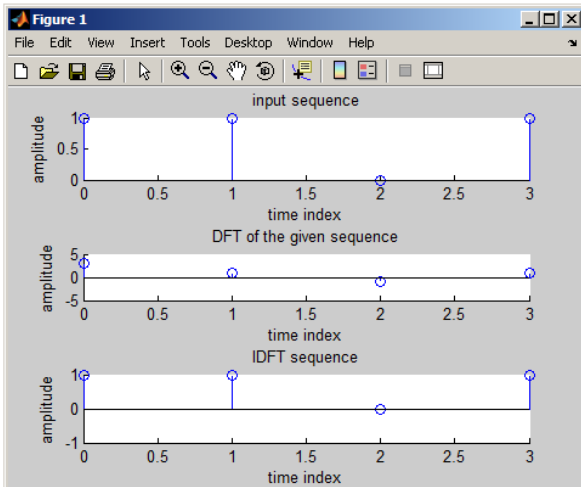
enter the input sequence[1 1 0 1]

```
3.0000      1.0000      -1.0000 - 0.0000i      1.0000
```

ixk =

```
1.0000 - 0.0000i      1.0000      -0.0000 + 0.0000i      1.0000 + 0.0000i
```

>>



EXPERIMENT No 7

DFT AND IDFT OF A SEQUENCE

AIM:

DFT and IDFT of a given sequence

TOOLS REQUIRED:

5. Mat lab software
6. Personal computer

PROGRAM:

```
clc;
close
all; clear
all;
xn=input('Enter the sequence x(n)'); %Get the sequence from user
ln=length(xn); %find the length of the sequence
xk=zeros(1,ln); %initilise an array of same size as that of input sequence
ixk=zeros(1,ln); %initilise an array of same size as that of input
sequence

%code block to find the DFT of the sequence
%-----
for k=0:ln-1
    for n=0:ln-1
        1
        xk(k+1)=xk(k+1)+(xn(n+1))*exp((-
        i)*2*pi*k*n/ln); end
    end
%-----

%code block to plot the input sequence
%-----
t=0:ln-1;
subplot(311);
stem(t,xn);
ylabel ('Amplitude');
xlabel ('Time
```

```

title('Input Sequence');
%-----
magnitude=abs(xk);           % Find the magnitudes of individual DFT points

%code block to plot the magnitude response
%-----
t=0:ln-1;
subplot(312);
stem(t,magnitude);
ylabel ('Amplitude');
xlabel ('K');
title('Magnitude Response');
%-----
% Code block to find the IDFT of the sequence
%-----
for n=0:ln-1
    for k=0:ln-1
        ixk(n+1)=ixk(n+1)+(xk(k+1)*exp(i*2*pi*k*n/ln)); end
    end
end
ixk=ixk/ln
;
%-----
%code block to plot the input sequence
t=0:ln-1;
subplot(313);
stem(t,ixk);ylabel ('Amplitude');xlabel ('Time Index');title('IDFT sequence');

```


OUTPUT AND WAVEFORMS:

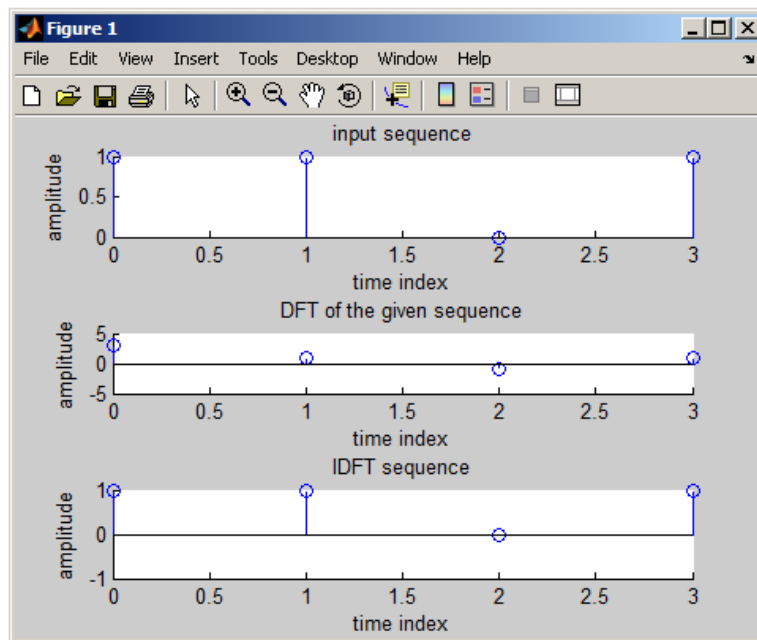
```
enter the input sequence[1 1 0 1]
```

```
3.0000      1.0000      -1.0000 - 0.0000i      1.0000
```

```
ixk =
```

```
1.0000 - 0.0000i      1.0000      -0.0000 + 0.0000i      1.0000 + 0.0000i
```

```
>>
```



Implementation of Linear convolution using DFT (Overlap-add and Overlap-Save methods)

```

a)Over lap add method clc
clear all close
all
x=[1 2 3 4 5 6 7 8 9 3 5 6 7];
h=[2 2 1];
% x=input('enter x');
% h=input('enter h');
% L=input('enter L')
M=length(h) lx=length(x)
L=5;
r=rem(lx,L);
x1=[x zeros(1,L-r)];
lx1=length(x1);
nr=length(x1)/L; h1=[h
zeros(1,L-1)]; for k=1:nr
    M1(k,:)=x1(((k-1)*L+1):k*L);
    M2(k,:)= [M1(k,:) zeros(1,M-1)];
    M3(k,:)=ifft(fft(M2(k,:)).*fft(h1)); M4(k,:)= [zeros(1,(k-1)*L) M3(k,:)
zeros(1,(nr-k)*L)];
end
y=sum(M4)

```

```

(b) Over lap save method clc;
clear all;
x=input('Enter 1st sequence X(n)= '); h=input('Enter 2nd
sequence H(n)= '); L=input('Enter length of each block L = ');

```

```

% Code to plot X(n) subplot
(2,2,1); stem(x);
stem(x,'blue'); xlabel ('n>');
ylabel ('Amplitude----->');
title('X(n)');

```

```

%Code to plot H(n)
subplot (2,2,2); stem(h);
stem(h,'black'); xlabel ('n
----->');
ylabel ('Amplitude----->');
title(' H(n)');

```

```

% Code to perform Convolution using Overlap Save Method

```

```

M=length(h);
lx=length(x);
r=rem(lx,L);
x1=[x zeros(1,L-r)];
nr=(length(x1))/L; h1=[h
zeros(1,L-1)]; for k=1:nr
    Ma(k,:)=x1(((k-1)*L+1):k*L)
    if k==1
        Ma1(k,:)=[zeros(1,M-1) Ma(k,:)];
    else
        Ma1(k,:)=[Ma(k-1,(L-M+2):L) Ma(k,:)];
    end
    Ma2(k,:)=ifft(fft(Ma1(k,:)).*fft(h1));
end
Ma3=Ma2(:,M:(L+M-1)); y1=Ma3';
y=y1(:)

% Representation of the Convoled Signal subplot (2,2,3:4);
stem(y,'red'); xlabel ('n
----->');
ylabel ('Amplitude----->');
title ('Convolved Signal');

```

EXPERIMENT-8

Implementation of Decimation-in-time radix-2 FFT algorithm

AIM:

FFT of a sequence using DIT-FFT method

TOOLS REQUIRED:

Mat lab software

Personal computer

% Direct computation of FFT

```
x=[1 1 0 0];
```

```
N=4;
```

```
y=fft(x,N);
```

```
stem(abs(y));
```

```
ylabel ('Amplitude');
```

```
xlabel ('N');
```

```
title('Magnitude Response');
```

%Matlab Program for FFT using DIT algorithm

```
clc; clear all; close all;
```

```
x=input('enter x[n]:');
```

```
N=length(x);
```

```
levels=nextpow2(N);
```

```
xn=[x,zeros(1,(2^levels)-
```

```
N)]; x=bitrevorder(xn)
```

```
N=length(xn);
```

```
tw=cos(2*pi*(1/N)*(0:N/2-1))-j*sin(2*pi*(1/N)*(0:N/2-1)); for level=1:levels;
```

```
    L=2^level;
```

```
    twlvl=tw(1:N/L:N/2)
```

```
    ; for k=0:L:N-L;
```

```
        for n=0:L/2-1;
```

```

A=x(n+k+1);
B=x(n+k+(L/2)+1)*twlv1(n+
1); x(n+k+1)=A+B;
x(n+k+(L/2)+1)=A-B;
end
end
x
end
XK
=x
n=0:N-1;
subplot(2,2,1);stem(n,xn);title('x(n)');xlabel('n');ylabel('Amplitude');
subplot(2,2,2);stem(n,real(XK));title('Real part of X(K)');xlabel('n');ylabel('Amplitude');
subplot(2,2,3);stem(n,imag(XK));title('Imag part of X(K)');xlabel('n');ylabel('Amplitude');

```

OUTPUT AND WAVEFORM:

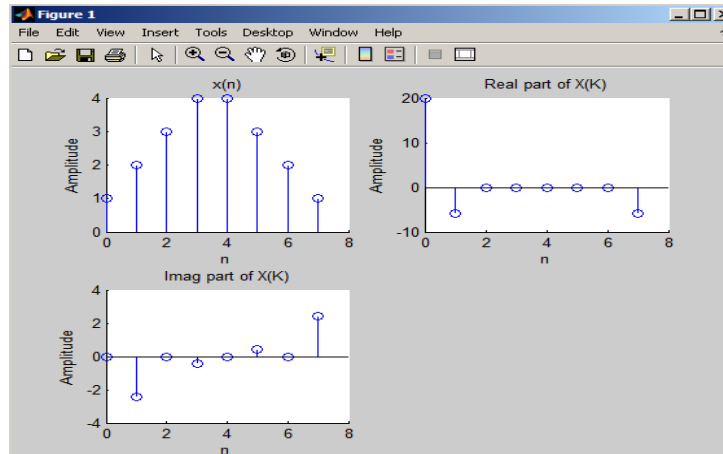
enter x[n]:[1 2 3 4 4 3 2 1]

XK =

20.0000 -5.8284 - 2.4142i 0 -0.1716 - 0.4142i 0 -0.1716 + 0.4142i

0 -5.8284 + 2.4142i

1



EXPERIMENT-9

Implementation of Decimation-in-frequency radix-2 FFT algorithm

AIM:

Implementation of Decimation-in-frequency radix-2 FFT algorithm

Tools Required:

Mat lab software

Personal computer

PROGRAM:

```
Clc;
Clear
all;
Close
all;
DECIMATION IN FREQUENCY [DIF] ALGORITHM

function q=dif(x)

t=nextpow2(length(x));           %Calculate the ndearest exponent of 2 j=[x
zeros(1,(2^t)-length(x))] ;% zero padding

N=length(j);                    % Length of padded structure
S=log2(N);                      % stages
for stage=S:-1:1 a=1;
b=1+2^(stage-1);                %Initialise a and b for each stage n=0;
while( n<=2^(stage-1)-1        && a<=N && b<=N)
    l=(n).*(2^(S+1-stage))./2;
    e=exp((-1i)*2*pi*l/(16));    % Twiddle factor
    y=j(a)+j(b);
    z=(j(a)-j(b)).*e;           % Butterfly structure j(a)=y;
    j(b)=z;
```

```

a=a+1; % Increment a,b and n
b=b+1;
n=n+1;
if (stage==1) % Discontinuity in the butterfly
structure
    if(rem(1,a)==1) % in a particular stage
        a=a+2^(stage-1);
        b=b+2^(stage-1); n=0;
    end
end
end

if(stage~=1) if(rem(a,2^(stage-1))==1)
    a=a+2^(stage-1); b=b+2^(stage-1);
    n=0;
    end
end
end
end

end

j=bitrevorder(j); % Bit reverse the output sequence disp(j);
q=j;

```

Implementation of IIR digital filter using Chebyshev (Type I and II) method

To Design a Chebyshev-I High pass filter for the given specifications using Matlab.

```
% To design a chebyshev-1 Hihpass filter for the given
specifications clf;
alphap=input('passband attenuation in db='); %passband attenuation in db
alphas=input('stopband attenuation in db=')% stopband attenuation in db
wp=.3*pi;% passband frequency in rad
ws=.2*pi;% stopband frequency in rad
%order and cutoff frequency of the filter
[n,wn]=cheblord(wp/pi,ws/pi,alphap,alphas);
%system function of the filter
[b,a]=cheby1(n,alphap,wn,'high');
w=0:0.01:pi;
[h,ph]=freqz(b,a,w);
m=20*log10(abs(h))
;
```

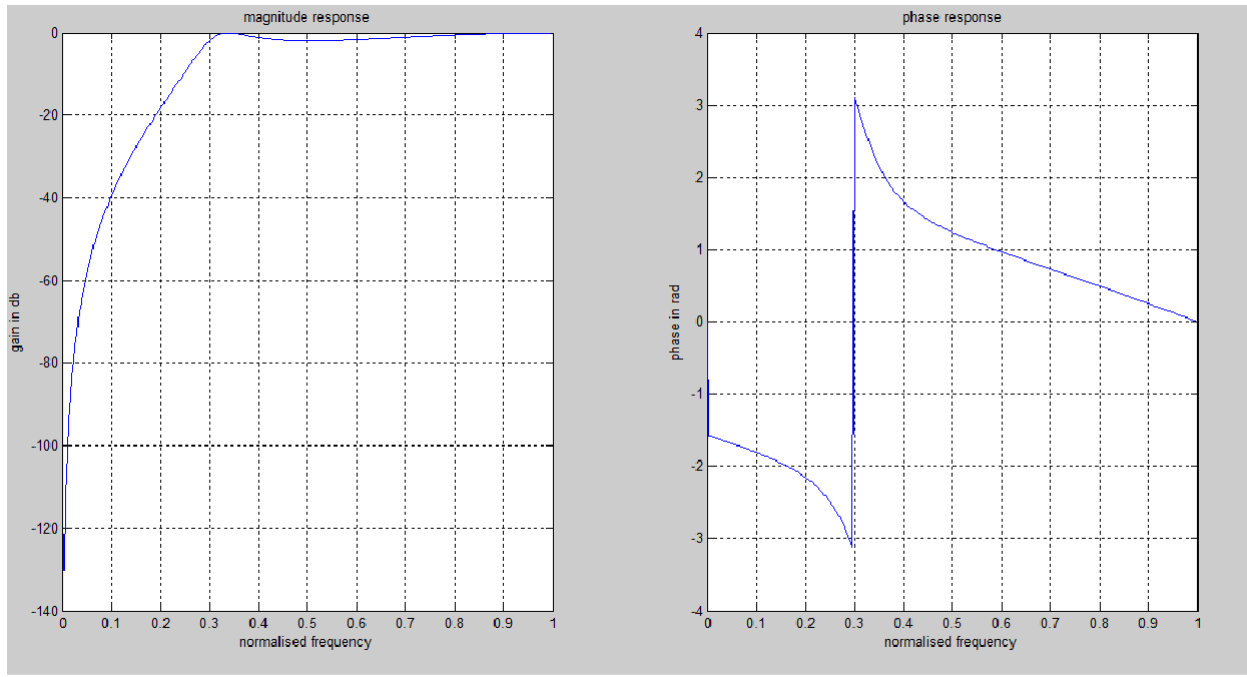


```
an=angle(h);
subplot(1,2,1);
plot(ph/pi,m);
grid;
    xlabel('normalised frequency');
    ylabel('gain in db');
    title('magnitude response');
subplot(1,2,2);
plot(ph/pi,an);
grid;
    xlabel('normalised frequency');
    ylabel('phase in rad');
    title('phase response'); disp(b);
disp(a);
```

INPUTS:

passband attenuation in db=1
stopband attenuation in db=15

OUTPUT WAVE FORMS:



B=0.2790 -0.8371 0.8371 -0.2790
A=1.0000 -0.7794 0.5677 0.1150

EXPERIMENT-10

Implementation of FIR digital filter using window (Rectangular, Hamming, Hanning, Bartlett) methods

AIM:

To Write a Matlab program of FIR Low pass and high pass filter using rectangular, Hanning Hamming, Blackman and Kaiser window..

TOOL:

MATLAB Software 9.0

PROGRAM:

```
%MATLAB program of FIR Low pass filter using Hanning %Hamming, Blackman and Kaiser
window clf;
wc=.5*pi;
N=25;
w=0:0.1:pi
;
b=fir1(N,wc/pi,blackman(N+1))
; h=freqz(b,1,w);
subplot(3,2,1)
plot(w/pi,abs(h))
grid;xlabel('normalised frequency');
ylabel('magnitude in dB')
title('FIR LPF USING BLACKMAN WINDOW')
b=fir1(N,wc/pi,hamming(N+1))
; h=freqz(b,1,w);
```

```

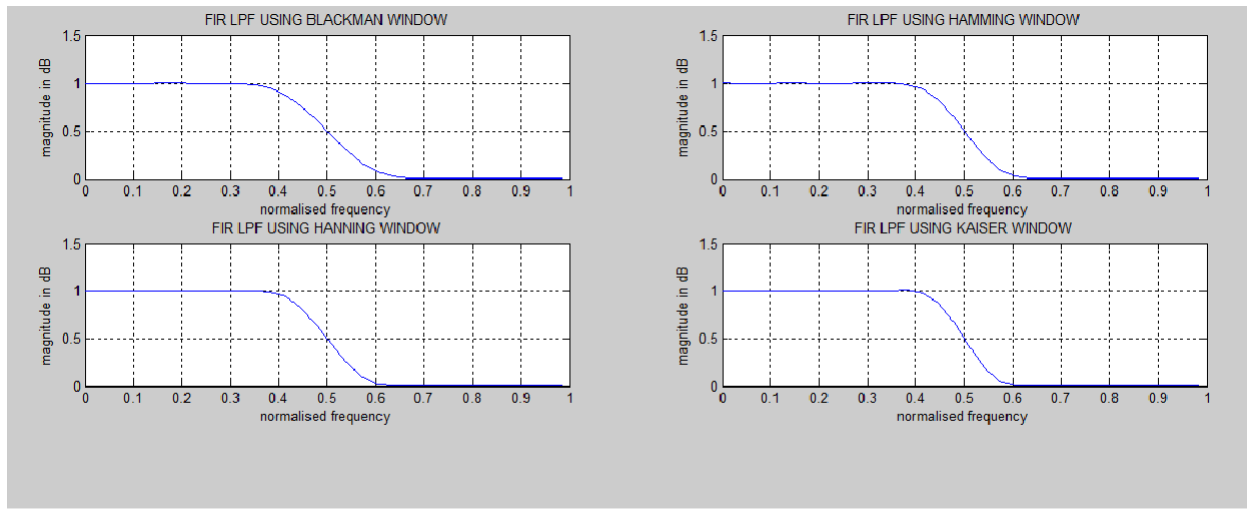
subplot(3,2,2)
plot(w/pi,abs(h));
grid;
xlabel('normalised frequency');
ylabel('magnitude in dB')
title('FIR LPF USING HAMMING WINDOW')

b=fir1(N,wc/pi,hanning(N+1))
; h=freqz(b,1,w);
subplot(3,2,3)
plot(w/pi,abs(h));
grid;
xlabel('normalised frequency');
ylabel('magnitude in dB')
title('FIR LPF USING HANNING WINDOW')

b=fir1(N,wc/pi,kaiser(N+1,3.5))
; h=freqz(b,1,w);
subplot(3,2,4)
plot(w/pi,abs(h));
grid;
xlabel('normalised frequency');
ylabel('magnitude in dB')
title('FIR LPF USING KAISER WINDOW')

```

OUTPUT WAVEFORMS:



Result :

PROGRAM:

```
%FIR Filter design window
techniques clc;

clear all;

close all;

rp=input('enter passband ripple');
rs=input('enter the stopband ripple');

fp=input('enter passband freq');
fs=input('enter stopband freq');
f=input('enter sampling freq ');
beta=input('enter beta value');

wp=2*fp/f; ws=2*fs/f;

num=-20*log10(sqrt(rp*rs))-
13; dem=14.6*(fs-fp)/f;
```

```

n=ceil(num/dem);
n1=n+1; if(rem(n,2)~=0) n1=n; n=n-
1; end
c=input('enter your choice of window function 1. rectangular 2. triangular 3.kaiser: \n ');
if(c==1) y=rectwin(n1);
    disp('Rectangular window filter response');

if (c==2) y=triang(n1);
    disp('Triangular window filter response');
end
if(c==3) y=kaiser(n1,beta);
    disp('kaiser window filter response');
end
%HPF
b=fir1(n,wp,'high',y);
[h,o]=freqz(b,1,256)
;
m=20*log10(abs(h))
; plot(o/pi,m);
title('HPF');
ylabel('Gain in dB-->');
xlabel('(b) Normalized frequency-->');

```

INPUT:

```

enter passband ripple:0.02
enter the stopband ripple:0.01
enter passband freq:1000 enter

```

stopband freq:1500 enter

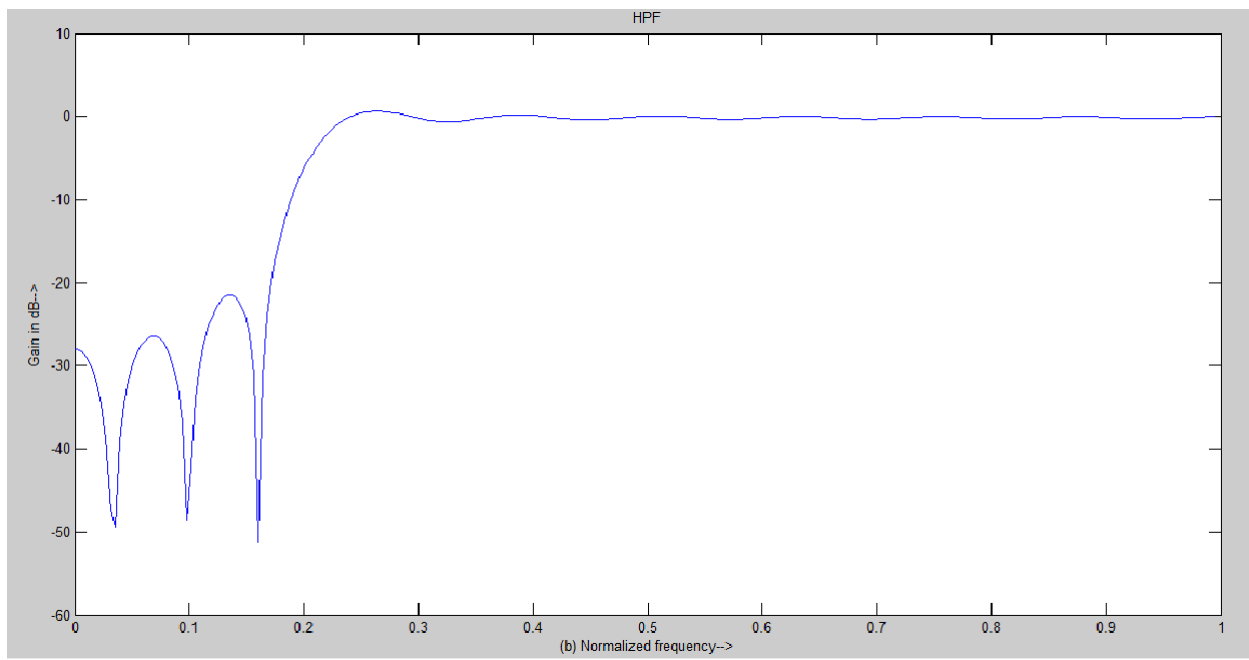
sampling freq: 10000 enter

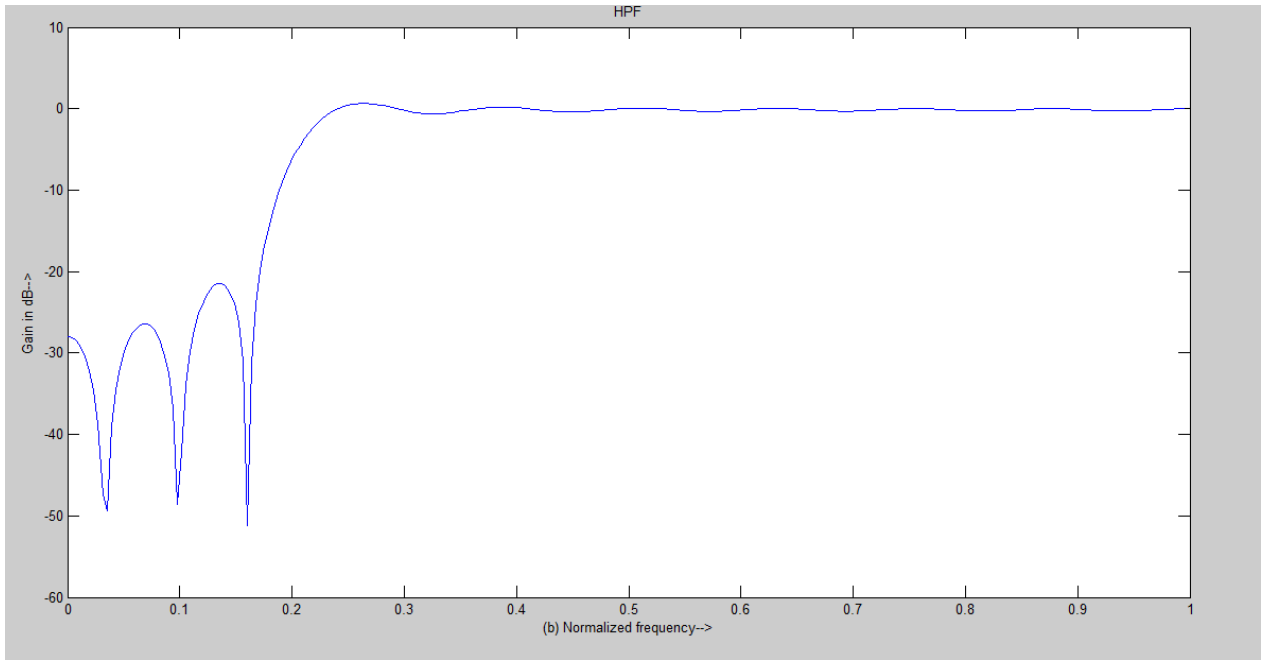
beta value:5

OUTPUT WAVEFORM:

enter your choice of window function 1. rectangular 2. triangular 3.kaiser: 2

Triangular window filter response

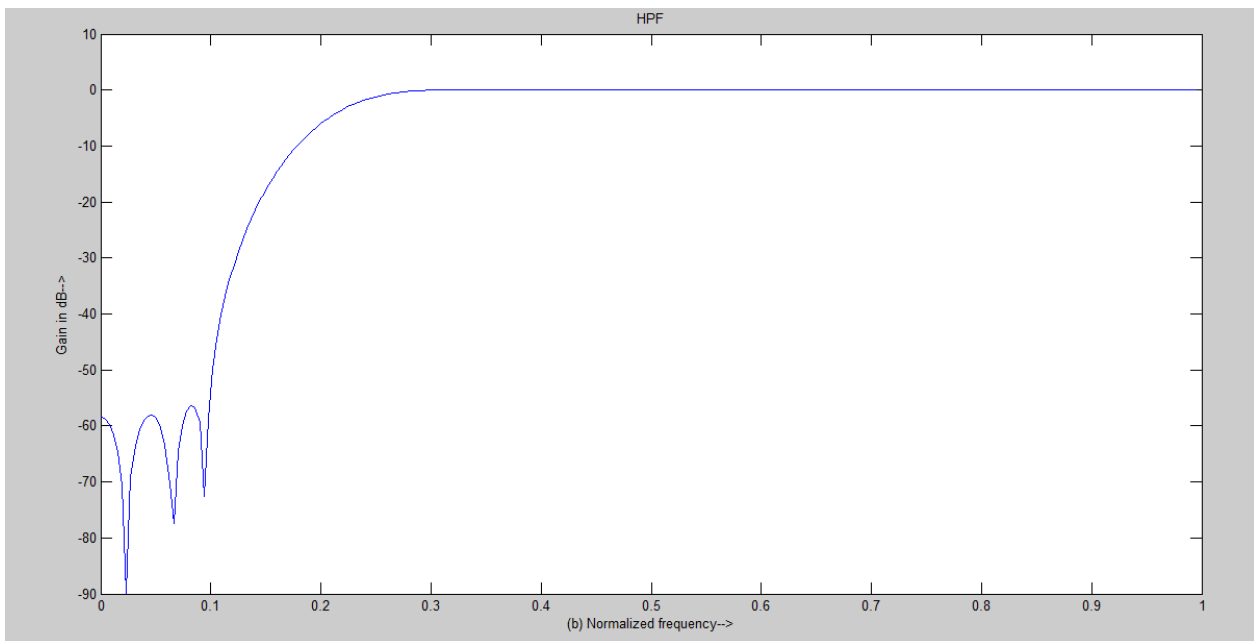




enter your choice of window function 1. rectangular 2. triangular 3.kaiser:

3

kaiser window filter response



EXPERIMENT-11

Implementation of FIR digital filter using frequency sampling method

AIM:

Decimation by factor D

TOOLS REQUIRED:

Mat lab software

Personal computer

PROGRAM:

```
N= 64; % FFT length = filter length
np = floor(N/2) + 1; % number of independent frequency points n = 0:np-1;
w = n*2*pi/N; % frequency vector
M = sin(n*pi/(np-1)); % some desired magnitude response
D = M.*exp(-1i*(N-1)/2*w); % desired complex frequency response (linear phase)
D = [D,conj(D(N-np+1:-1:2))]; % append redundant points for IFFT h = ifft(D);%
compute impulse response % should be very close to 0
max(abs(imag(h)))

h = real(h); % remove numerical inaccuracies

% check result
[H,w2] = freqz(h,1,4*N);
plot(w/2/pi,abs(D(1:np)),',w2/2/pi,abs(H))
```

OUTPUT AND WAVEFORM

Implementation of optimum equiripple FIR digital filter using window methods

AIM:

To Write a Matlab program of FIR Low pass and high pass filter using rectangular, Hanning Hamming, Blackman and Kaiser window.

.

TOOL:

MATLAB Software 9.0

PROGRAM:

```
%MATLAB program of FIR Low pass filter using Hanning  
%Hamming, Blackman and Kaiser
```

```
window clf;
```

```
wc=.5*pi;
```

```
N=25;
```

```
w=0:0.1:pi
```

```
;
```

```
b=fir1(N,wc/pi,blackman(N+1))
```

```
; h=freqz(b,1,w);
```

```
subplot(3,2,1)
```

```
plot(w/pi,abs(h))
```

```
grid;xlabel('normalised frequency');
```

```
ylabel('magnitude in dB')
```

```
title('FIR LPF USING BLACKMAN WINDOW')
```

```
b=fir1(N,wc/pi,hamming(N+1))
```

```
; h=freqz(b,1,w);
```

```
subplot(3,2,2)
```

```
plot(w/pi,abs(h));
```

```
grid;
```

```
xlabel('normalised frequency');
```

```
ylabel('magnitude in dB')
```

```
title('FIR LPF USING HAMMING WINDOW')
```

```
b=fir1(N,wc/pi,hanning(N+1))
```

```
; h=freqz(b,1,w);
```

```
subplot(3,2,3)
```

```
plot(w/pi,abs(h));
```

```

grid;

xlabel('normalised frequency');

ylabel('magnitude in dB')

title('FIR LPF USING HANNING WINDOW')

b=fir1(N,wc/pi,kaiser(N+1,3.5))

; h=freqz(b,1,w);

subplot(3,2,4)

plot(w/pi,abs(h));

grid;

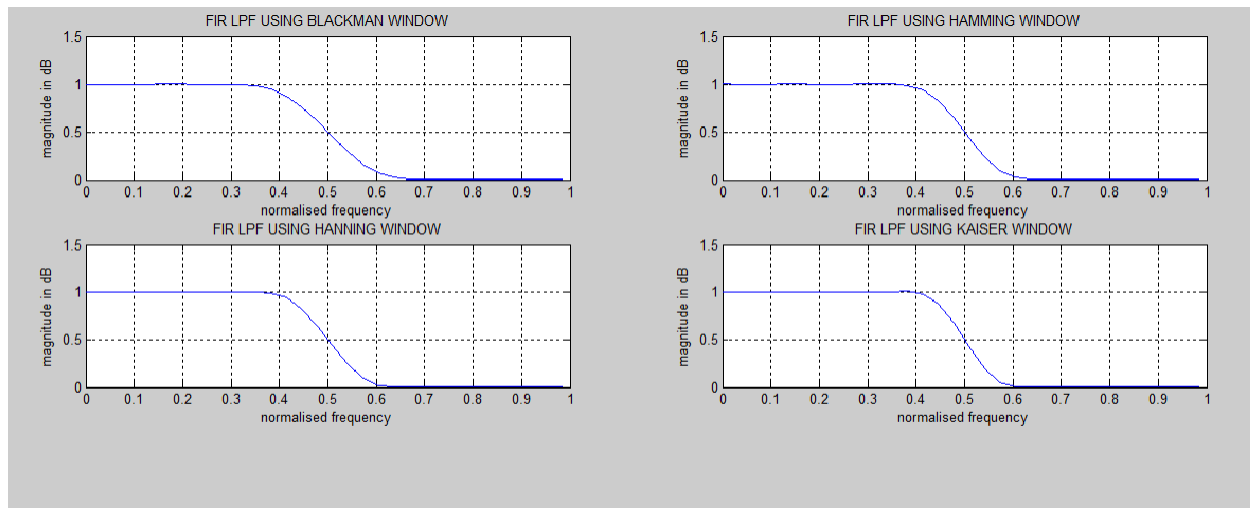
xlabel('normalised frequency');

ylabel('magnitude in dB')

title('FIR LPF USING KAISER WINDOW')

```

OUTPUT WAVEFORMS:



PROGRAM:

```
%FIR Filter design window
techniques clc;

clear all;
close all;

rp=input('enter passband ripple');
rs=input('enter the stopband ripple');
fp=input('enter passband freq');
fs=input('enter stopband freq');
f=input('enter sampling freq ');
beta=input('enter beta value');
wp=2*fp/f; ws=2*fs/f;
num=-20*log10(sqrt(rp*rs))-
13; dem=14.6*(fs-fp)/f;
n=ceil(num/dem);
n1=n+1; if(rem(n,2)~=0) n1=n; n=n-
1; end
c=input('enter your choice of window function 1. rectangular 2. triangular 3.kaiser: \n ');
if(c==1) y=rectwin(n1);
    disp('Rectangular window filter response');
end
if (c==2) y=triang(n1);
    disp('Triangular window filter response');
```

```

end
if(c==3) y=kaiser(n1,beta);
    disp('kaiser window filter response');
end
%HPF
b=fir1(n,wp,'high',y);
[h,o]=freqz(b,1,256)
;
m=20*log10(abs(h))
; plot(o/pi,m);
title('HPF');
ylabel('Gain in dB-->');
xlabel('(b) Normalized frequency-->');

```

INPUT:

```

enter passband ripple:0.02
enter the stopband ripple:0.01
enter passband freq:1000 enter
stopband freq:1500 enter
sampling freq: 10000 enter
beta value:5

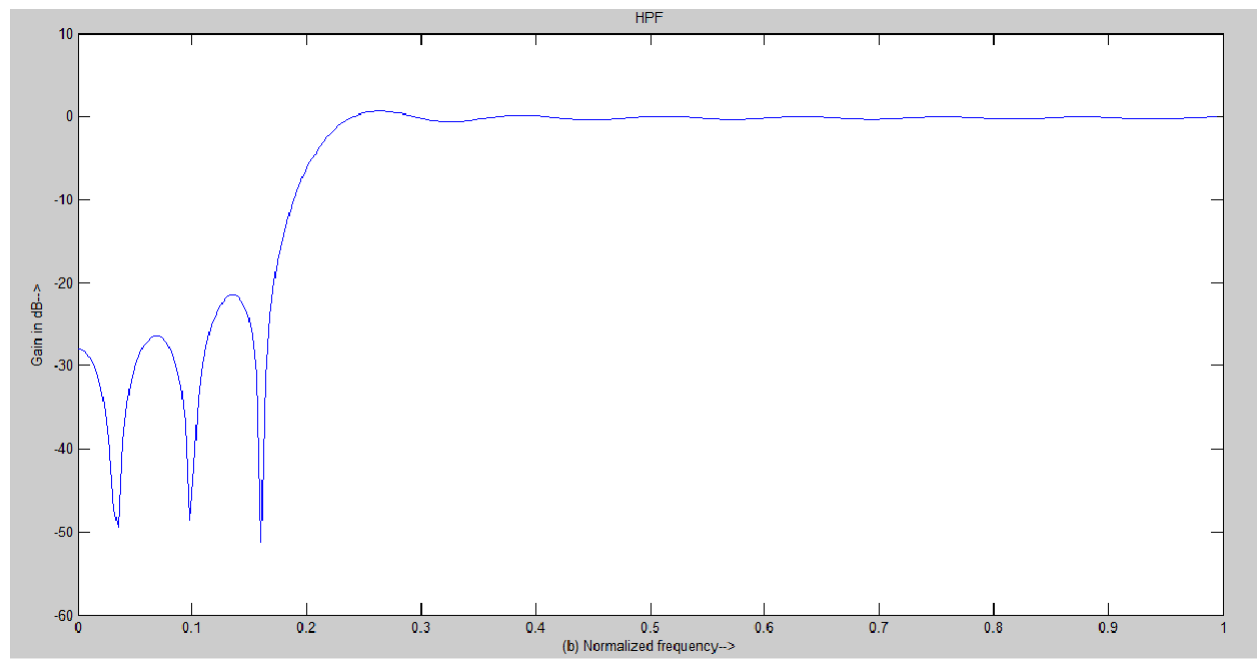
```

OUTPUT

enter your choice of window function 1. rectangular 2. triangular 3.kaiser:

2

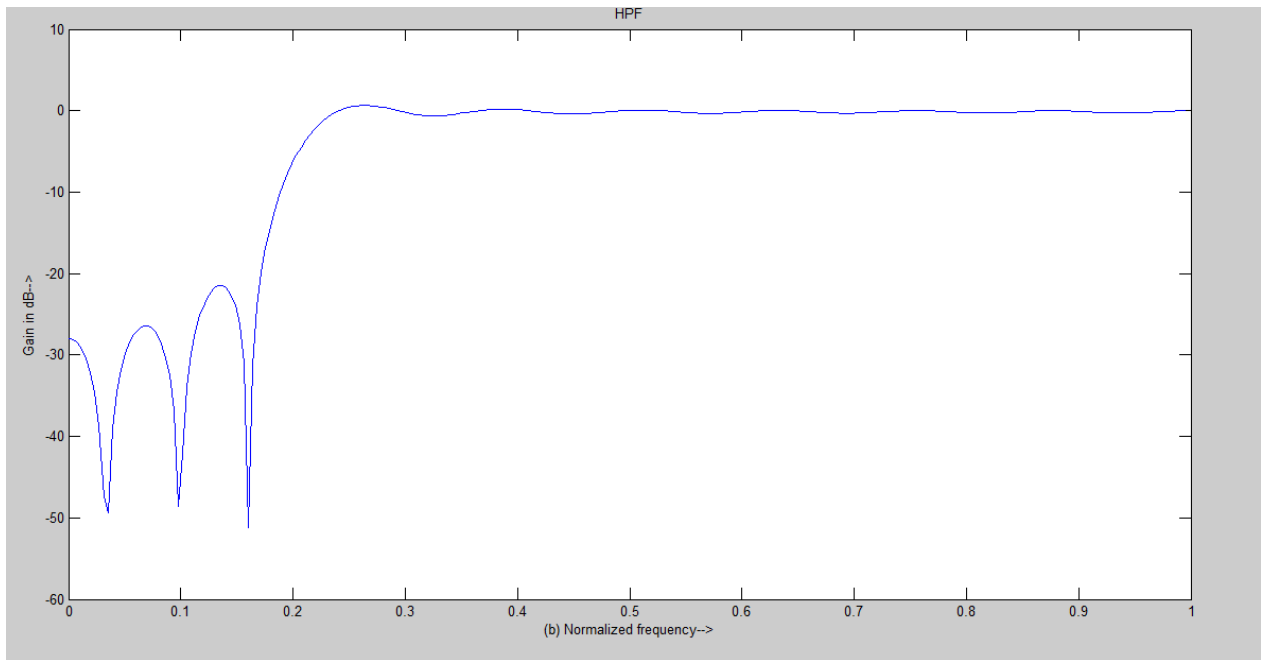
Triangular window filter response



enter your choice of window function 1. rectangular 2. triangular 3.kaiser:

1

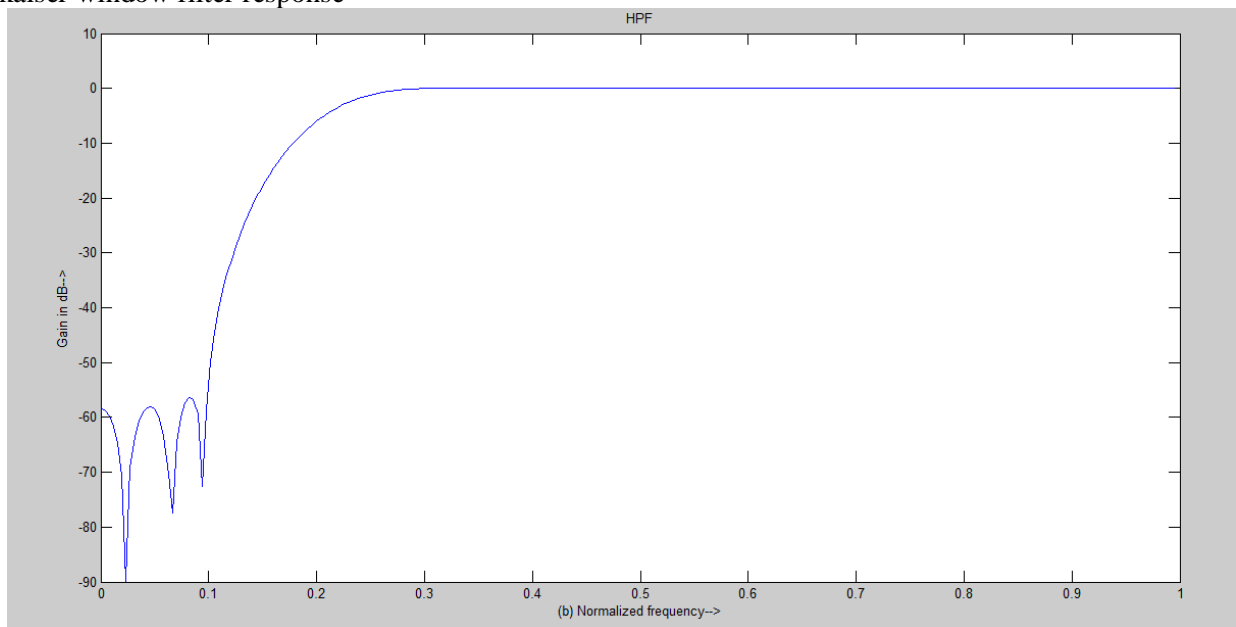
Rectangular window filter response



enter your choice of window function 1. rectangular 2. triangular 3.kaiser:

3

kaiser window filter response



9. DTMF Tone Generation and Detection Using Goertzel Algorithm

AI

M: DTMF Tone Generation and Detection Using Goertzel Algorithm

TOOLS REQUIRED:

Mat lab software

Personal computer

PROGRAM:

```
close all;clear all

% DTMF tone generator

fs=8000;

t=[0:1:204]/fs;

x=zeros(1,length(t));

x(1)=1;

y852=filter([0 sin(2*pi*852/fs) ],[1 -2*cos(2*pi*852/fs) 1],x);

y1209=filter([0 sin(2*pi*1209/fs) ],[1 -2*cos(2*pi*1209/fs)

1],x); y7=y852+y1209;

subplot(2,1,1);plot(t,y7);grid

ylabel('y(n) DTMF: number 7');

xlabel('time (second)');

title('signal tone number 7')

Ak=2*abs(fft(y7))/length(y7);Ak(1)=Ak(1)/

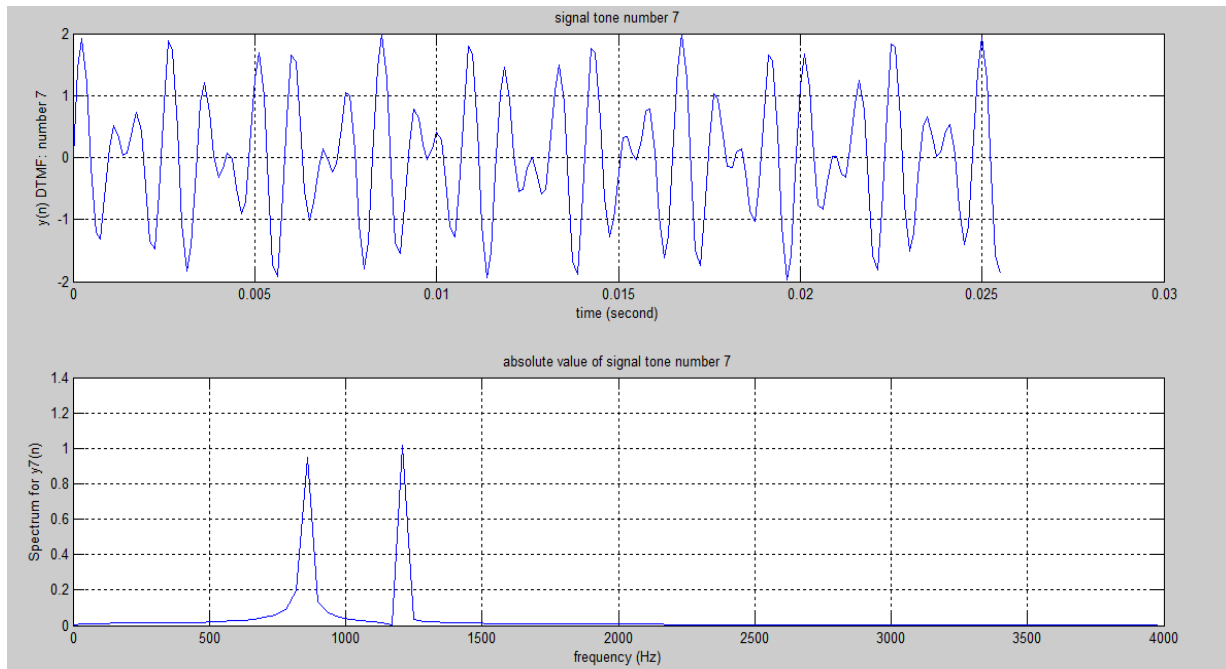
2; f=[0:1:(length(y7)-1)/2]*fs/length(y7);

subplot(2,1,2);plot(f,Ak(1:(length(y7)+1)/2));grid
```



```
ylabel('Spectrum for y7(n)');  
xlabel('frequency (Hz)');  
title('absolute value of signal tone number 7')
```

OUTPUT WAVEFORM:



EXPERIMENT – 12

Implementation of sampling rate conversion by decimation, interpolation and a rational factor using MATLAB

AIM:

Sampling rate conversion by a factor I/D

TOOLS REQUIRED:

Mat lab software

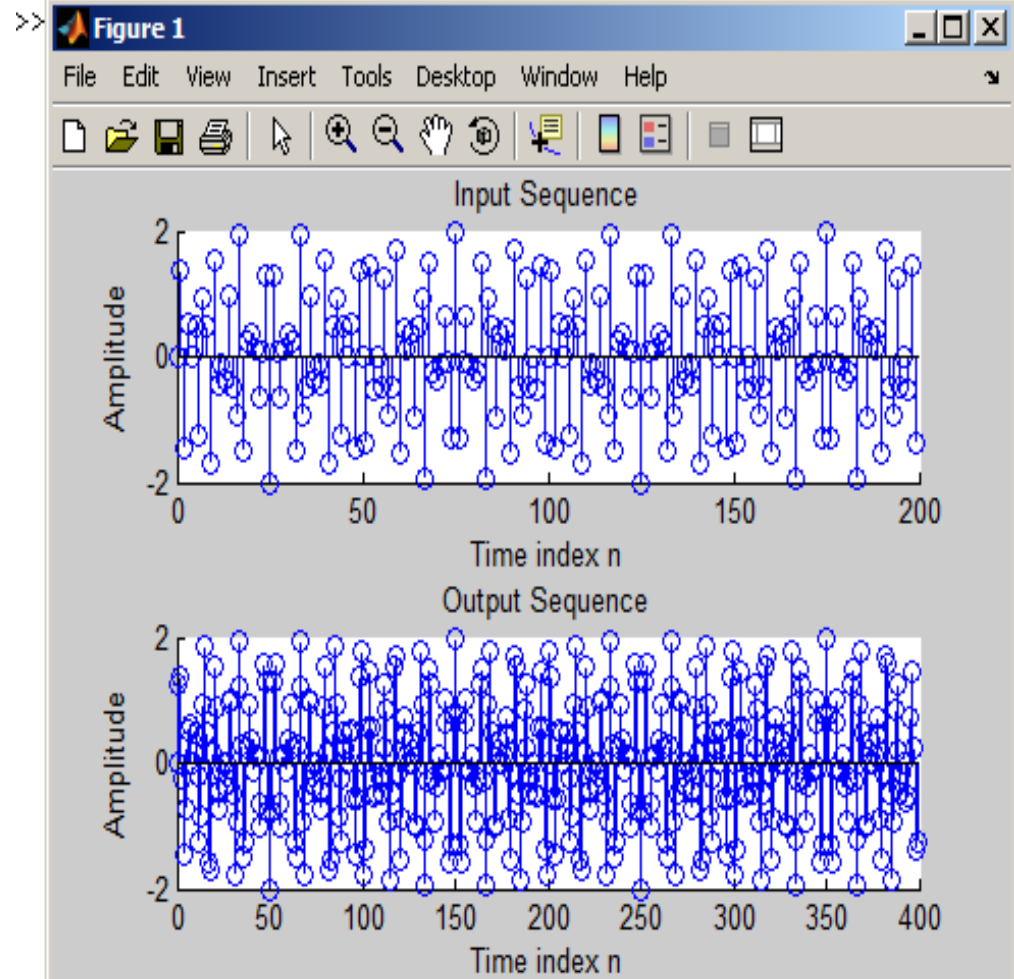
Personal computer

PROGRAM:

```
clc; close all; clear all;
L = input('Enter Up-sampling factor :');
M = input('Enter Down-sampling factor :');
N = input('Enter number of samples :');
n = 0:N-1;
x = sin(2*pi*0.43*n) +
sin(2*pi*0.31*n); y = resample(x,L,M);
subplot(2,1,1); stem(n,x(1:N));
axis([0 29 -2.2 2.2]);
title('Input Sequence');
xlabel('Time index n');
ylabel('Amplitude'); subplot(2,1,2);
m = 0:(N*L/M)-1;
stem(m,y(1:N*L/M));
axis([0 (N*L/M)-1 -2.2
2.2]);
title('Output Sequence');
xlabel('Time index n'); ylabel('Amplitude');
```

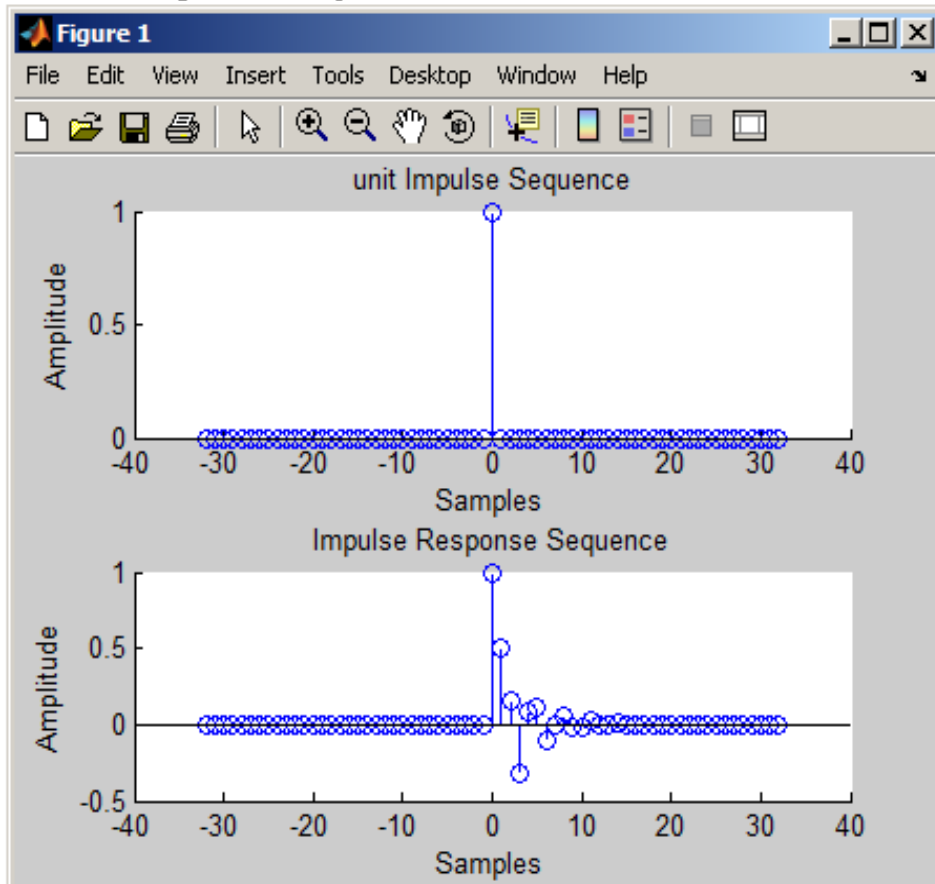
OUTPUT AND WAVEFORM:

```
Enter Up-sampling factor :6  
Enter Down-sampling factor :3  
Enter number of samples :200
```



```
enter the coefficients of numerator polynomial= [1 1 0.9]
enter the coefficients of denominator polynomial= [1 0.5 0.5]
enter the length of sequence= 32
```

>>



INTRODUCTION TO DSP PROCESSORS

EXPERIEMENTS

(Using DSP Kit)

A signal can be defined as a function that conveys information, generally about the state or behavior of a physical system. There are two basic types of signals viz Analog (continuous time signals which are defined along a continuum of times) and Digital (discrete-time).

Remarkably, under reasonable constraints, a continuous time signal can be adequately represented by samples, obtaining discrete time signals. Thus digital signal processing is an ideal choice for anyone who needs the performance advantage of digital manipulation along with today's analog reality.

Hence a processor which is designed to perform the special operations (digital manipulations) on the digital signal within very less time can be called as a Digital signal processor. The difference between a DSP processor, conventional microprocessor and a microcontroller are listed below.

Microprocessor: or General Purpose Processor such as Intel xx86 or Motorola 680xx

Family

Contains - only CPU

-No RAM

-No ROM

-No I/O ports

-No Timer

MICROCONTROLLER

Such as 8051 family Contains - CPU

- RAM
- ROM
- I/O ports
- Timer &
- Interrupt circuitry

Some Micro Controllers also contain A/D, D/A and Flash Memory

DSP PROCESSORS

Such as Texas instruments and Analog Devices Contains

- CPU
- RAM
- ROM
- I/O ports
- Timer

Optimized for

- fast arithmetic
- Extended precision
- Dual operand fetch
- Zero overhead loop
- Circular buffering

The basic features of a DSP Processor are

Feature	Use
Fast-Multiply accumulate	Most DSP algorithms, including filtering, transforms, etc. are multiplication- intensive
Multiple – access memory architecture	Many data-intensive DSP operations require reading a program instruction and multiple data items during each instruction cycle for best performance
Specialized addressing modes	Efficient handling of data arrays and first-in, first-out buffers in memory
Specialized program control	Efficient control of loops for many iterative DSP algorithms. Fast interrupt handling for frequent I/O operations.
On-chip peripherals and I/O interfaces	On-chip peripherals like A/D converters allow for small low cost system designs. Similarly I/O interfaces tailored for common peripherals allow clean interfaces to off-chip I/O devices.

A digital signal processor (DSP) is an integrated circuit designed for high-speed data manipulations, and is used in audio, communications, image manipulation, and other data-acquisition and data-control applications. The microprocessors used in personal computers are optimized for tasks involving data movement and inequality testing. The typical applications requiring such capabilities are word processing, database management, spread sheets, etc. When it comes to mathematical computations the traditional microprocessor are deficient particularly where real-time performance is required. Digital signal processors are microprocessors optimized for basic mathematical calculations such as additions and multiplications.

FIXED VERSUS FLOATING POINT:

Digital Signal Processing can be divided into two categories, fixed point and floating point which refer to the format used to store and manipulate numbers within the devices. Fixed point DSPs usually represent each number with a minimum of 16 bits, although a different length can be used. There are four common ways that these 2^{16} i.e., 65,536 possible bit patterns can represent a number. In unsigned integer, the stored number can take on any integer value from 0 to 65,535, signed integer uses two's complement to include negative numbers from - 32,768 to 32,767. With unsigned fraction notation, the 65,536 levels are spread uniformly between 0 and 1 and the signed fraction format allows negative numbers, equally spaced between -1 and 1. The floating point DSPs typically use a minimum of 32 bits to store each value. This results in many more bit patterns than for fixed point, 2^{32} i.e., 4,294,967,296 to be exact. All floating point DSPs can also handle fixed point numbers, a necessity to implement counters, loops, and signals coming from the ADC and going to the DAC. However, this doesn't mean that fixed point math will be carried out as quickly as the floating point operations; it depends on the internal architecture.

C VERSUS ASSEMBLY:

DSPs are programmed in the same languages as other scientific and engineering applications, usually assembly or C. Programs written in assembly can execute faster, while programs written in C are easier to develop and maintain. In traditional applications, such as programs run on PCs and mainframes, C is almost always the first choice. If assembly is used at all, it is restricted to short subroutines that must run with the utmost speed.

HOW FAST ARE DSPS?

The primary reason for using a DSP instead of a traditional microprocessor is speed: the ability to move samples into the device and carry out the needed mathematical operations, and output the processed data. The usual way of specifying the fastness of a DSP is: fixed point

systems are often quoted in MIPS (million integer operations per second). Likewise, floating point devices can be specified in MFLOPS (million floating point operations per second).

TMS320 FAMILY:

The Texas Instruments TMS320 family of DSP devices covers a wide range, from a 16-bit fixed-point device to a single-chip parallel-processor device. In the past, DSPs were used only in specialized applications. Now they are in many mass-market consumer products that are continuously entering new market segments. The Texas Instruments TMS320 family of DSP devices and their typical applications are mentioned below.

C1x, C2x, C2xx, C5x, and C54x:

The width of the data bus on these devices is 16 bits. All have modified Harvard architectures. They have been used in toys, hard disk drives, modems, cellular phones, and active car suspensions.

C3x:

The width of the data bus in the C3x series is 32 bits. Because of the reasonable cost and floating-point performance, these are suitable for many applications. These include almost any filters, analyzers, hi-fi systems, voice-mail, imaging, bar-code readers, motor control, 3D graphics, or scientific processing.

C4x:

This range is designed for parallel processing. The C4x devices have a 32-bit data bus and are floating-point. They have an optimized on-chip communication channel, which enables a number of them to be put together to form a parallel-processing cluster. The C4x range devices have been used in virtual reality, image recognition, telecom routing, and parallel-processing systems.

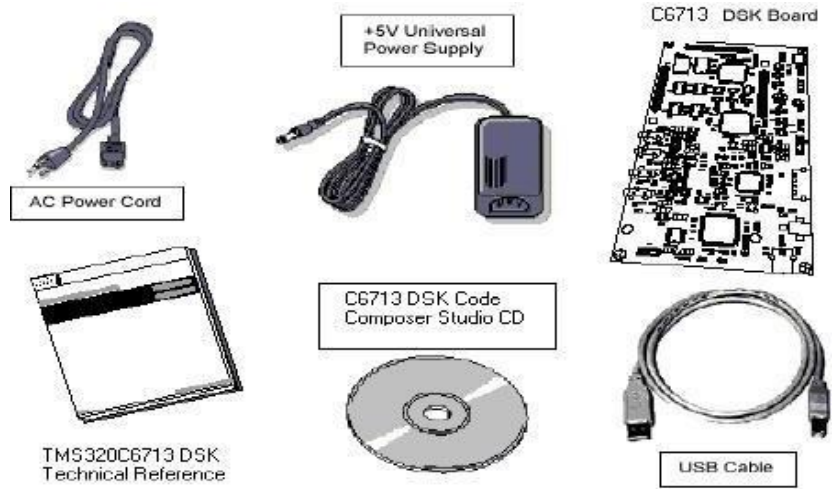
C6x:

The C6x devices feature Velocity, an advanced very long instruction word (VLIW) architecture developed by Texas Instruments. Eight functional units, including two multipliers and six

arithmetic logic units (ALUs), provide 1600 MIPS of cost-effective performance. The C6x DSPs are optimized for multi-channel, multifunction applications, including wireless base stations, pooled modems, remote-access servers, digital subscriber loop systems, cable modems, and multi-

INTRODUCTION TO TMS 320 C6713 DSK

The high-performance board features the TMS320C6713 floating-point DSP. Capable of performing 1350 million floating point operations per second, the C6713 DSK the most powerful DSK development board.

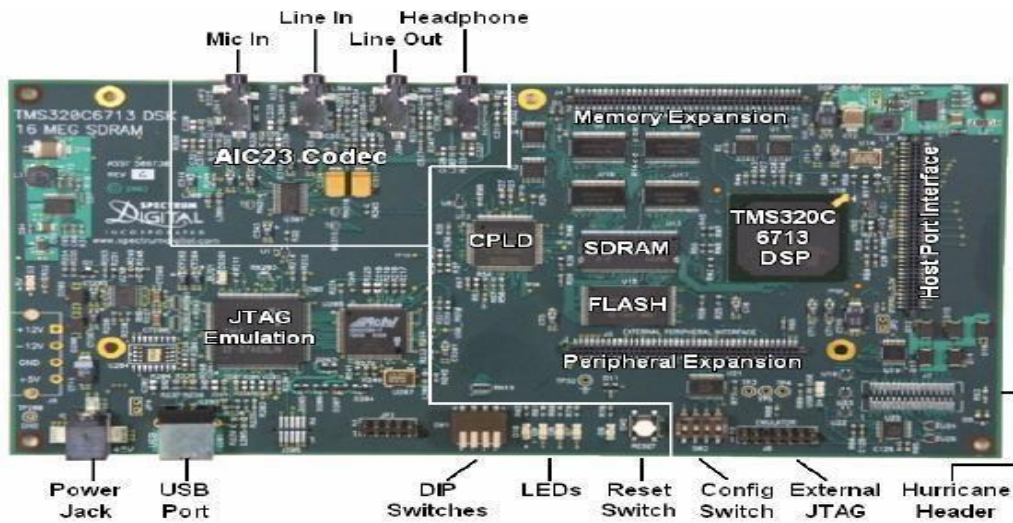
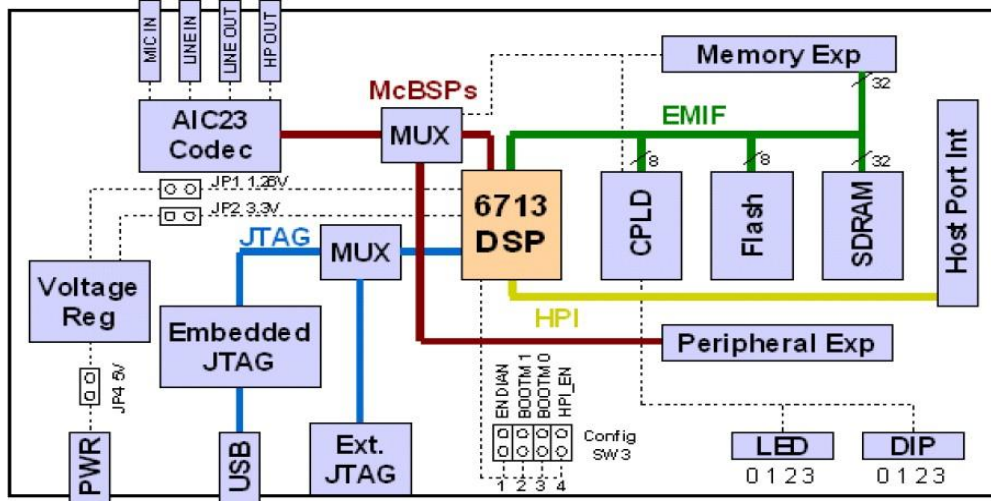


The DSK is USB port interfaced platform that allows to efficiently develop and test applications for the C6713. With extensive host PC and target DSP software support, the DSK provides ease-of-use and capabilities that are attractive to DSP engineers. The 6713 DSP Starter Kit (DSK) is a low-cost platform which lets customers evaluate and develop applications for the Texas Instruments C67X DSP family. The primary features of the DSK are:

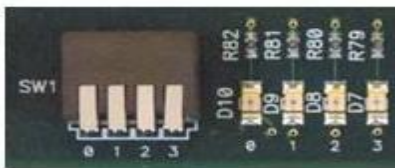
1. 225 MHz TMS320C6713 Floating Point DSP
2. AIC23 Stereo Codec
3. Four Position User DIP Switch and Four User LEDs
4. On-board Flash and SDRAM

TI's Code Composer Studio development tools are bundled with the 6713DSK providing the user with an industrial-strength integrated development environment for C and assembly programming. Code Composer Studio communicates with the DSP using an on-board JTAG emulator through a USB interface. The TMS320C6713 DSP is the heart of the system. It is a core member of Texas Instruments' C64X line of fixed point DSPs whose distinguishing features are an extremely high performance 225MHz VLIW DSP core and 256Kbytes of internal memory. On-chip peripherals include a 32-bit external memory interface (EMIF) with integrated SDRAM controller, 2 multi-channel buffered serial ports (McBSPs), two on-board timers and an enhanced DMA controller (EDMA). The 6713 represents the high end of TI's C6700 floating point DSP line both in terms of computational performance and on-chip resources.

The 6713 has a significant amount of internal memory so many applications will have all code and data on-chip. External accesses are done through the EMIF which can connect to both synchronous and asynchronous memories. The EMIF signals are also brought out to standard TI expansion bus connectors so additional functionality can be added on daughter card modules. DSPs are frequently used in audio processing applications so the DSK includes an on-board codec called the AIC23. Codec stands for coder/decoder, the job of the AIC23 is to code analog input samples into a digital format for the DSP to process, then decode data coming out of the DSP to generate the processed analog output. Digital data is sent to and from the codec on McBSP1.



The DSK has 4 light emitting diodes (LEDs) and 4 DIP switches that allow users to interact with programs through simple LED displays and user input on the switches. Many of the included examples make use of these user interfaces Options.

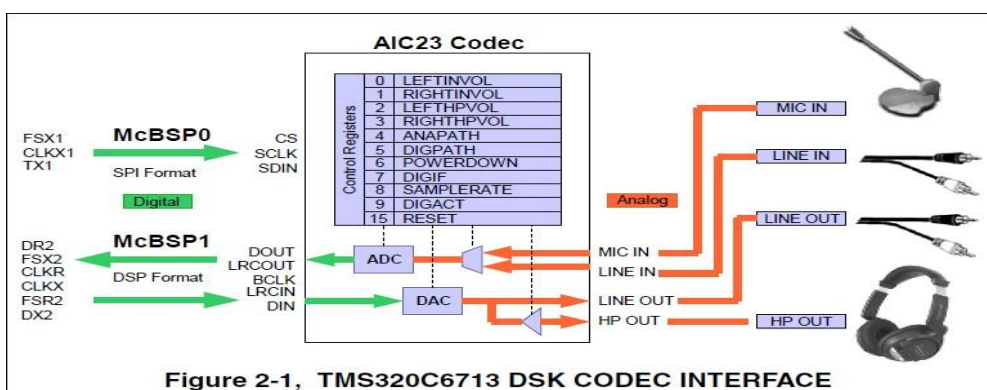


The DSK implements the logic necessary to tie board components together in a programmable logic device called a CPLD. In addition to random glue logic, the CPLD implements a set of 4 software programmable registers that can be used to access the on-board LEDs and DIP switches as well as control the daughter card interface.

AIC23 Codec

The DSK uses a Texas Instruments AIC23 (part #TLV320AIC23) stereo codec for input and output of audio signals. The codec samples analog signals on the microphone or line inputs and converts them into digital data so it can be processed by the DSP. When the DSP is finished with the data it uses the codec to convert the samples back into analog signals on the line and headphone outputs so the user can hear the output.

The codec communicates using two serial channels, one to control the codec's internal configuration registers and one to send and receive digital audio samples. McBSP0 is used as the unidirectional control channel. It should be programmed to send a 16-bit control word to the AIC23 in SPI format. The top 7 bits of the control word should specify the register to be modified and the lower 9 should contain the register value. The control channel is only used when configuring the codec, it is generally idle when audio data is being transmitted, McBSP1 is used as the bi-directional data channel. All audio data flows through the data channel. Many data formats are supported based on the three variables of sample width, clock signal source and serial data format. The DSK examples generally use a 16-bit sample width with the codec in master mode so it generates the frame sync and bit clocks at the correct sample rate without effort on the DSP side. The preferred serial format is DSP mode which is designed specifically to operate with the McBSP ports on TI DSPs.



DSK hardware installation

- Shut down and power off the PC
- Connect the supplied USB port cable to the board
- Connect the other end of the cable to the USB port of PC
- Plug the other end of the power cable into a power outlet
- Plug the power cable into the board
- The user LEDs should flash several times to indicate board is operational
- When you connect your DSK through USB for the first time on a Windows

Loaded PC the new hardware found wizard will come up. **So, Install the drivers** (The CCS CD contains the require drivers for C6713 DSK).

TROUBLESHOOTING DSK CONNECTIVITY

If Code Composer Studio IDE fails to configure your port correctly, perform the following steps:

Test the USB port by running DSK Port test from the start menu

Use Start->Programs->Texas Instruments->Code Composer Studio-> Code Composer Studio

C6713 DSK Tools -> C6713 DSK Diagnostic Utilities

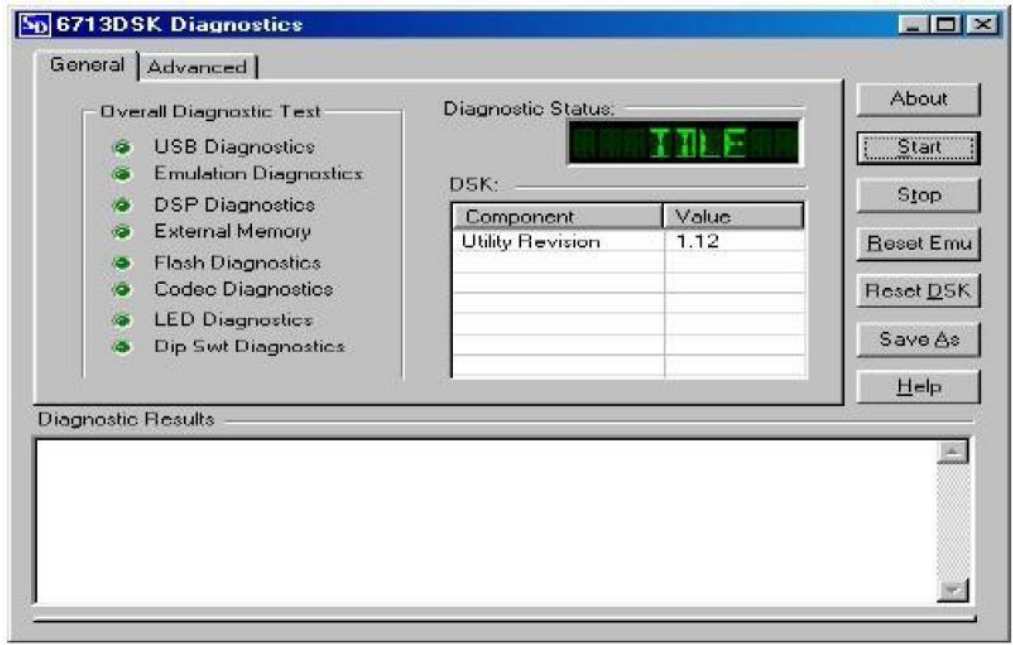
The below Screen will appear

Select 6713 DSK Diagnostic Utility Icon from Desktop, The Screen Look like as below

Select **Start** Option

Utility Program will test the board

After testing Diagnostic Status you will get **PASS**



EXTRA Syllabus

Implementation of DFT USING TMS 320C6713 Kit

a) Sine wave generation using lookup table with values generated from MATLAB

AIM:

Computation of N-point DFT of a Sequence Using DSK Code composer studio

EQUIPMENTS:

TMS 320C6713 Kit.

RS232 Serial

Cable Power Cord

Operating System – Windows

XP Software – CCStudio_v3.1

THEORY:

In this program the Discrete Fourier Transform (DFT) of a sequence $x[n]$ is generated by using the formula,

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-2\pi j k n / N}$$

Where, $X(k)$ □ DFT of sequence

N represents the sequence length and it is calculated by using the command 'length'. The DFT of any sequence is the powerful computational tool for performing frequency analysis of discrete-time signals.

PROGRAM:

```
#include <stdio.h>

#include <math.h>

int N,k,n,i;

float

pi=3.1416,sumre=0,sumim=0,out_real[8]={0.0},out_imag[8]={0.0}; int

x[32];

void main(void)

{

printf("enter the length of the sequence\n");

scanf("%d",&N);

printf("\nenter the sequence\n");
```

```

for(i=0;i<N;i++)

scanf("%d",&x[i]);

for(k=0;k<N;k++)

{

sumre=0;

sumim=0;

for(n=0;n<N;n++)

)

{

sumre=sumre+x[n]*cos(2*pi*k*n/N)

; sumim=sumim-

x[n]*sin(2*pi*k*n/N);

}

out_real[k]=sumre;

out_imag[k]=sumim;

printf("DFT of the sequence:\n");

printf("x[%d]=\t%f\t+\t%fi\n",k,out_real[k],out_imag[k]);

}

}

```

PROCEDURE:

- Open code composer studio, make sure the dsp kit is turned on.
- Start a new project using ‘project-new ‘ pull down menu, save it in a separate directory(d:11951a0xxx) with name **dft**.

- Write the program and save it as dft.c
- Add the source files dft.c to the project using ‘project->add files to project’ pull down menu.
- Add the linker command file hello.cmd.
(path: c:ccstudio_v3.1\tutorial\dsk6713\hello1\hello.cmd)
- Add the run time support library file rts6700.lib.
(path: c:ccstudio_v3.1\c6000\cgtools\lib\rts6700.lib)
- Compile the program using the ‘project-compile’ pull down menu
- Build the program using the ‘project-build’ pull down menu
- Load the program (dft.out) in program memory of dsp chip using the ‘file-load program’ pull down menu.
- Debug-> run
- To view output graphically select view ->graph ->time and frequency.

OUTPUT AND WAVEFORM:

enter the length of the sequence

4

enter the sequence

1 2 3 4

DFT of the sequence:

$x[0]= 10.000000 + 0.000000i$

DFT of the sequence:

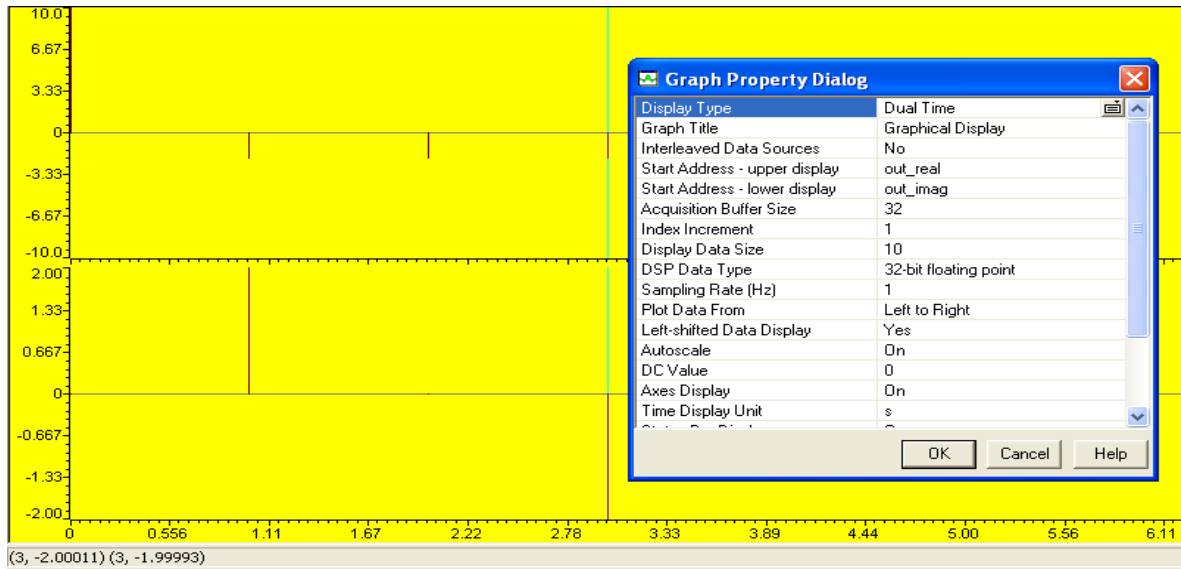
$x[1]= -1.999963 + 2.000022i$

DFT of the sequence:

$x[2]= -2.000000 + 0.000059i$

DFT of the sequence:

$$x[3] = -2.000108 + -1.999934i$$



Empty Space for

Calculations Insert Graph

Sheet (Normal)

GENERATION OF SINE WAVE USING C6713 DSK

AIM:

To generate a real time sinewave using TMS320C6713 DSK

EQUIPMENTS:

TMS 320C6713 Kit.

RS232 Serial

Cable Power Cord

Operating System – Windows

XP Software – CCStudio_v3.1

PROCEDURE:

1. Connect CRO to the LINE OUT socket.
2. Now switch ON the DSK and bring up Code Composer Studio on PC
3. Create a new project with name sinewave.pjt
4. From File menu->New->DSP/BIOS Configuration->Select dsk6713.cdb and save it as “sinewave.cdb”
5. Add sinewave.cdb to the current project
6. Create a new source file and save it as sinewave.c
7. Add the source file sinewave.c to the project
8. Add the library file “dsk6713bsl.lib” to the project
(Path: C:\CCStudio\C6000\dsk6713\lib\dsk6713bsl.lib)
9. Copy files “dsk6713.h” and “dsk6713_aic23.h” to the Project folder
(Path: C:\CCStudio_v3.1\C6000\dsk6713\include)
10. Build (F7) and load the program to the DSP Chip (File->Load Program(.out file))
11. Run the program (F5)
12. Observe the waveform that appears on the CRO screen and cstudio simulator.

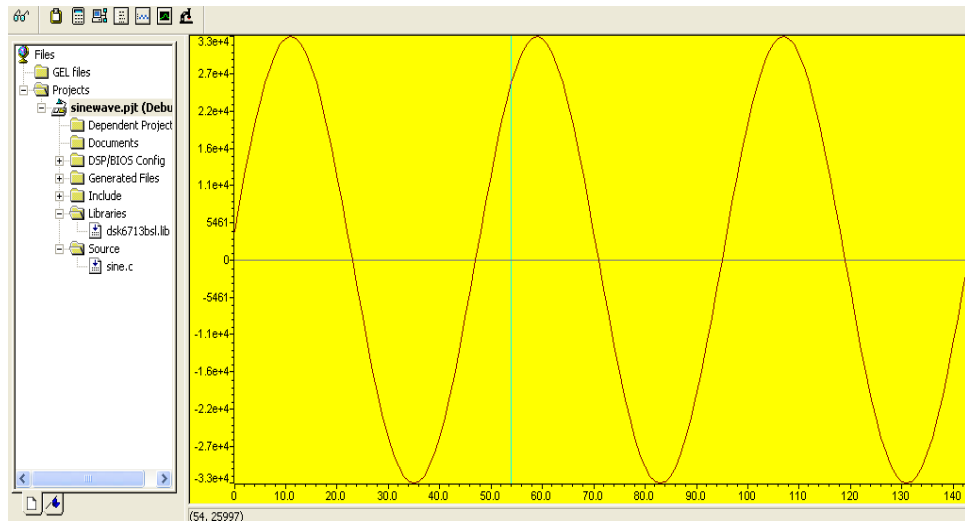
```

%% matlab code to generate the sine values of the look table
n=1:48;
x=sin(2*pi*n*1000/48000);
x1=round(x*2^15);
// c program for generation of sine wave using c6713 DSK
#include "sinewavecfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
short loop=0;
short gain=1;
Int16 outbuffer[256];
Const short BUFFERLENGTH=256; int i=0;
DSK6713_AIC23_Config
config={0x0017,0x0017,0x00d8,0x00d8,0x0011,0x0000,0x0000,0x0043,0x0081,0x0001};
Int16 sine_table[48]={4277,8481,12540,16384,19948,23170,25997,28378,30274,31651,32488,
32766,32488,31651,30274,28378,25997,23170,19948,16384,12540,8481,4277,0,-4277,-8481,
-12540,-16384,-19948,-23170,-25997,-28378,-30274,-31651,-32488,-32766,-32488,-31651,
-30274,-28378,-25997,-23170,-19948,-16384,-12540,-8481,-
4277,0}; Uint32 fs=DSK6713_AIC23_FREQ_48KHZ;
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
DSK6713_init();
hCodec=DSK6713_AIC23_openCodec(0,
&config); DSK6713_AIC23_setFreq(hCodec, fs);
while(1)
{
outbuffer[i]=sine_table[loop];
while(!DSK6713_AIC23_write(hCodec,
sine_table[loop])*gain); i++;
if(i==BUFFERLENGTH) i=0;
if(++loop>47) loop=0;
}
}

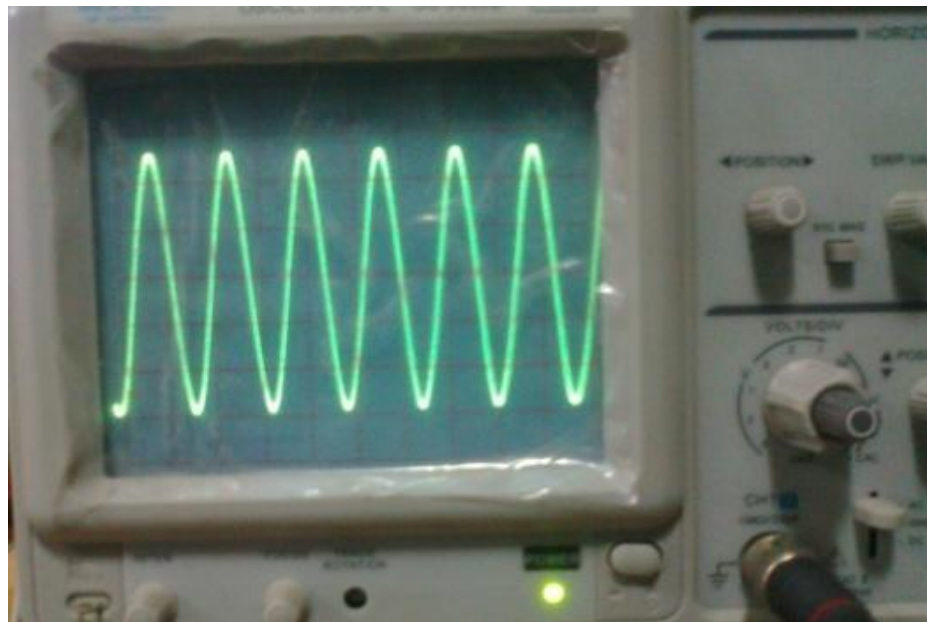
```

WAVEFORM:

a) output from code composer studio



b) output from the CRO



INSERT GRAPH SHEET (NORMAL)

14. IR and FIR Filter Implementation using DSP Kits

FIR FILTER USING RECTANGULAR WINDOW

AIM:

To generate a real time fir filter through Rectangular window using TMS320C6713 DSK

EQUIPMENTS:

TMS 320C6713 Kit.

RS232 Serial

Cable Power Cord

Operating System – Windows

XP Software – CCStudio_v3.1

PROCEDURE:

1. Connect CRO to the LINE OUT sockets.
2. Now switch ON the DSK and bring up Code Composer Studio on PC
3. Create a new project with name sinewave.pjt
4. From File menu->New->DSP/BIOS Configuration->Select dsk6713.cdb and save it as “firfilter.cdb”
5. Add firfilter.cdb to the current project
6. Create a new source file and save it as firfilter.c
7. Add the source file firfilter.c to the project
8. Add the library file “dsk6713bsl.lib” to the project
(Path: C:\CCStudio\C6000\dsk6713\lib\dsk6713bsl.lib)
9. Copy files “dsk6713.h” and “dsk6713_aic23.h” to the Project folder
(Path: C:\CCStudio_v3.1\C6000\dsk6713\include)
10. Build (F7) and load the program to the DSP Chip (File->Load Program(.out file))
11. Run the program (F5)
12. Observe the waveform that appears on the CRO screen and ccstudio simulator.

// c program for generation of fir filter using c6713 DSK

```
#include "firfiltercfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#include "stdio.h"
Float    filter_coeff[    ]={-0.020203,-0.016567,0.009656,0.027335,0.011411,-0.023194,-
0.033672,0.000000,0.043293,0.038657,-0.025105,-0.082004,-0.041842,0.115971,0.303048,
0.386435,0.303048,0.115971,-0.041842,-0.082004,-0.025105,0.038657,0.043293,0.000000,-
0.033672,-0.023194,0.011411,0.027335,0.009656,-0.016567,-0.020203};//FIR    Low    pass
Rectangular Filter pass band range 0-1500Hz
DSK6713_AIC23_Config
config={0x0017,0x0017,0x00d8,0x00d8,0x0011,0x0000,0x0000,0x0043,0x0081,0x0001};
void main()
{
    DSK6713_AIC23_CodecHandle hCodec;
    Uint32 l_input, r_input,l_output, r_output;
    DSK6713_init();
    hCodec = DSK6713_AIC23_openCodec(0,
    &config); DSK6713_AIC23_setFreq(hCodec, 1);
    while(1)
    {
        while(!DSK6713_AIC23_read(hCodec,
        &l_input));
        while(!DSK6713_AIC23_read(hCodec,
        &r_input));
        l_output=(Int16)FIR_FILTER(&filter_coeff
        ,l_input); r_output=l_output;
        while(!DSK6713_AIC23_write(hCodec,
        l_output));
        while(!DSK6713_AIC23_write(hCodec,
        r_output));
    }
    DSK6713_AIC23_closeCodec(hCodec);
```

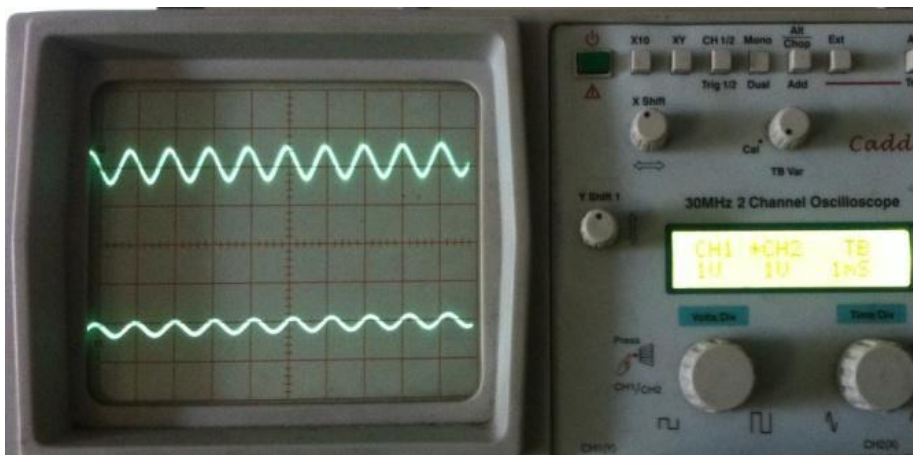
```

}
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
static short int in_buffer[100];
in_buffer[0] = x;
for(i=30;i>0;i--)
in_buffer[i] = in_buffer[i-
1]; for(i=0;i<32;i++)
output = output + h[i] * in_buffer[i];
return(output);
}

```

WAVEFORM:

- a) Waveforms of input and output from cro



- b) Function generator (input signal frequency at 1 KHz)



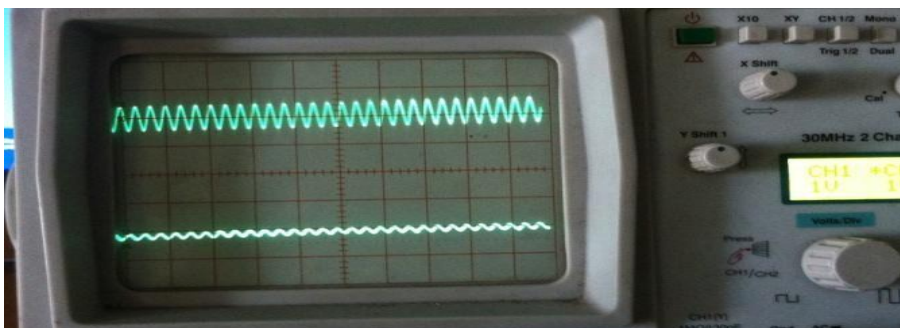
c) C6713 DSK



d) Function generator (input signal frequency at 1500Hz)



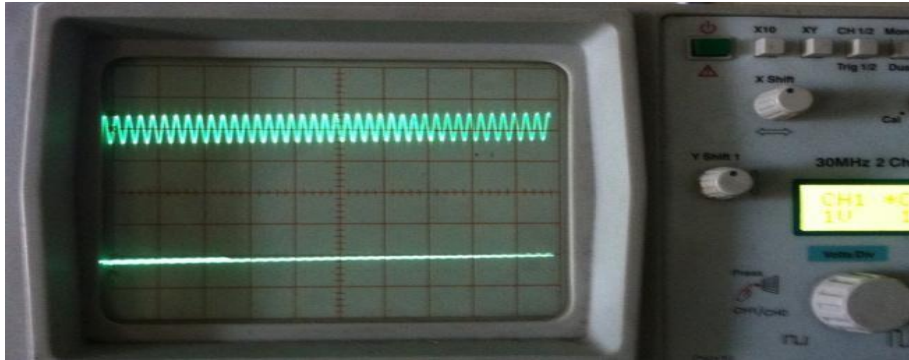
e) Waveforms of input and output from CRO (output signal attenuates)



d) Function generator (input signal frequency at 2000Hz)



a) Waveforms of input and output from CRO (output signal fully attenuated)



INSERT GRAPH SHEET (NORMAL)

15.FIR FILTER USING KAISER WINDOW (HIGH PASS)

AIM:

To generate a real time fir filter through Kaiser Window using TMS320C6713 DSK

EQUIPMENTS:

TMS 320C6713 Kit.

RS232 Serial

Cable Power Cord

Operating System – Windows

XP Software – CCStudio_v3.1

PROCEDURE:

1. Connect CRO to the LINE OUT sockets.
2. Now switch ON the DSK and bring up Code Composer Studio on PC
3. Create a new project with name sinewave.pjt
4. From File menu->New->DSP/BIOS Configuration->Select dsk6713.cdb and save it as “firfliter_kaiser.cdb”
5. Add firfilter.cdb to the current project
6. Create a new source file and save it as firfilter_kaiser.c
7. Add the source file firfilter_kaiser.c to the project
8. Add the library file “dsk6713bsl.lib” to the project
(Path: C:\CCStudio\C6000\dsk6713\lib\dsk6713bsl.lib)
9. Copy files “dsk6713.h” and “dsk6713_aic23.h” to the Project folder
(Path: C:\CCStudio_v3.1\C6000\dsk6713\include)
10. Build (F7) and load the program to the DSP Chip (File->Load Program (.out file))
11. Run the program (F5)
12. Observe the waveform that appears on the CRO screen and cstudio simulator.

```
// c program for generation of fir filter using c6713 DSK
```

```
#include
```

```
"firfilter_kaisercfg.h"
```

```
#include "dsk6713.h"
```

```
#include "dsk6713_aic23.h"
```

```
#include "stdio.h"
```

```
float filter_coeff[] = {0.000000, -0.000138, -0.000611, -0.001345, -0.001607, 0.000000, 0.004714,  
0.012033, 0.018287, 0.016731, 0.000000, -0.035687, -0.086763, -0.141588, -0.184011, 0.800005, -  
0.184011, -0.141588, -0.086763, -0.035687, 0.000000, 0.016731, 0.018287, 0.012033, 0.004714, -  
0.000000, -0.001607, -0.001345, -0.000611, -0.000138, 0.000000}; //FIR High pass Kaiser filter pass  
band range 800Hz-3.5KHz
```

```
void main()
```

```
{
```

```
DSK6713_AIC23_CodecHandle hCodec;
```

```
Uint32 l_input, r_input, l_output, r_output;
```

```
DSK6713_init();
```

```
hCodec = DSK6713_AIC23_openCodec(0,
```

```
&config); DSK6713_AIC23_setFreq(hCodec, 1);
```

```
while(1)
```

```
{
```

```
while(!DSK6713_AIC23_read(hCodec, &l_input));
```

```
while(!DSK6713_AIC23_read(hCodec, &r_input));
```

```
l_output = (Int16)FIR_FILTER(&filter_coeff, l_input);
```

```
r_output = l_output;
```

```
while(!DSK6713_AIC23_write(hCodec, l_output));
```

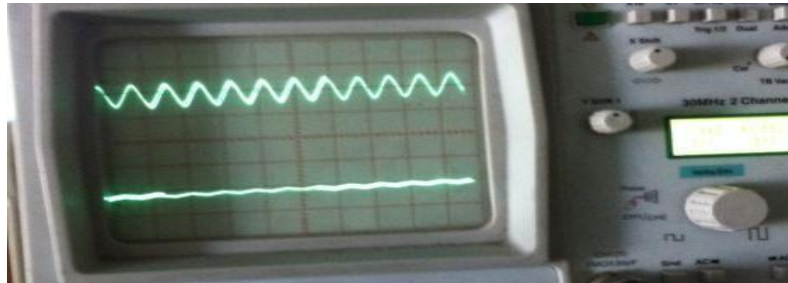
```

while(!DSK6713_AIC23_write(hCodec, r_output));
}
DSK6713_AIC23_closeCodec(hCodec);
}
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
static short int in_buffer[100];
in_buffer[0] = x;
for(i=30;i>0;i--)
in_buffer[i] = in_buffer[i-
1]; for(i=0;i<32;i++)
output = output + h[i] * in_buffer[i];
//output = x;
return(output);
}

```

WAVEFORM:

- a) Waveforms of input and output from CRO



- b) function generator (input signal frequency at 500Hz)



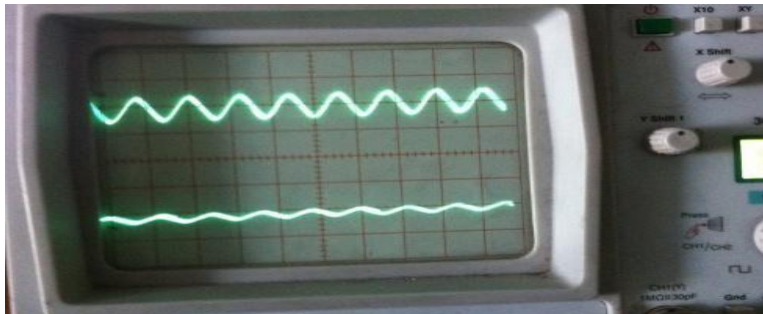
c) C6713 DSK



d) function generator (input signal frequency at 800Hz)



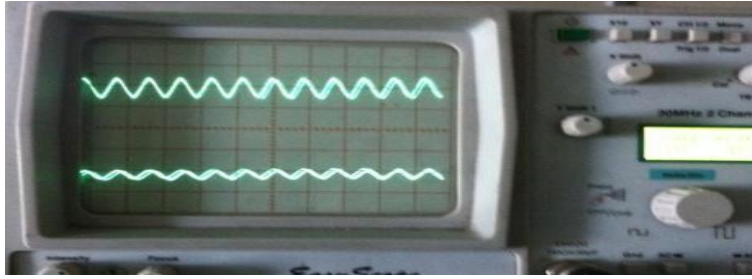
e) Waveforms of input and output from CRO



d) function generator (input signal frequency at 1.1kHz)



e) Waveforms of input and output from CRO



Insert Graph Sheet (Normal)

IIR filter using Butterworth Approximation (low pass)

AIM:

To generate a real time iir filter through Butterworth approximation using TMS320C6713 DSK

EQUIPMENTS:

TMS 320C6713 Kit.

RS232 Serial

Cable Power Cord

Operating System – Windows

XP Software – CCStudio_v3.1

PROCEDURE:

1. Connect CRO to the LINE OUT socket.
2. Now switch ON the DSK and bring up Code Composer Studio on PC
3. Create a new project with name sinewave.pjt
4. From File menu->New->DSP/BIOS Configuration->Select dsk6713.cdb and save it as “iirfliter.cdb”
5. Add firfilter.cdb to the current project
6. Create a new source file and save it as iirfilter.c
7. Add the source file iirfilter.c to the project
8. Add the library file “dsk6713bsl.lib” to the project

(Path: C:\CCStudio\C6000\dsk6713\lib\dsk6713bsl.lib)

9. Copy files “dsk6713.h” and “dsk6713_aic23.h” to the Project folder
(Path: C:\CCStudio_v3.1\C6000\dsk6713\include)
10. Build (F7) and load the program to the DSP Chip (File->Load Program(.out file))
11. Run the program (F5)
12. Observe the waveform that appears on the CRO screen and cccstudio simulator.

// c program for generation of iir filter using c6713 DSK

```
#include "iirfiltercfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#include "stdio.h"
const signed int filter_coeff [ ] = { 15241,15241,15241,32761,10161,7877};
//IIR_BUTERWORTH_LP FILTER pass band range 0-
8kHz DSK6713_AIC23_Config
config={0x0017,0x0017,0x00d8,0x00d8,0x0011,0x0000,0x0000,0x0043,0x0081,0x0001};
void main()
{
    DSK6713_AIC23_CodecHandle hCodec;
    Uint32 l_input, r_input,l_output, r_output;
    DSK6713_init();
    hCodec = DSK6713_AIC23_openCodec(0,
    &config); DSK6713_AIC23_setFreq(hCodec, 3);
    while(1)
    {
        while(!DSK6713_AIC23_read(hCodec, &l_input));
        while(!DSK6713_AIC23_read(hCodec, &r_input));
        l_output=IIR_FILTER(&filter_coeff ,l_input);
        r_output=l_output;
        while(!DSK6713_AIC23_write(hCodec, l_output));
        while(!DSK6713_AIC23_write(hCodec, r_output));
    }
}
```

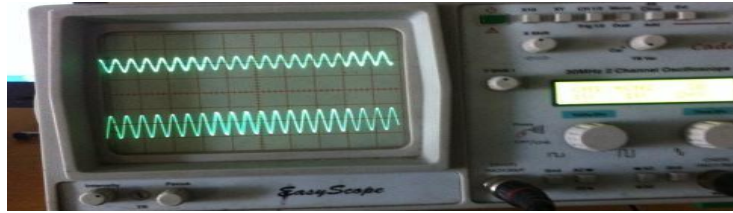
```

DSK6713_AIC23_closeCodec(hCodec);
}
signed int IIR_FILTER(const signed int * h, signed int x1)
{
    static    signed    int    x[6]    =
    {0,0,0,0,0,0}; static signed int y[6]
    = {0,0,0,0,0,0}; int temp=0;
    temp = (short int)x1;
    x[0] = (signed int) temp;
    temp = ( (int)h[0] *
    x[0]);
    temp += ( (int)h[1] * x[1]);
    temp += ( (int)h[1] * x[1]);
    temp += ( (int)h[2] * x[2]);
    temp -= ( (int)h[4] * y[1]);
    temp -= ( (int)h[4] * y[1]);
    temp -= ( (int)h[5] * y[2]);
    temp >>=15;
    if ( temp > 32767 )
    {
        temp = 32767;
    }
    else if ( temp < -32767)
    {
        temp = -32767;
    }
    y[0] = temp;
    y[2] = y[1];
    y[1] = y[0];
    x[2] = x[1];
    x[1] = x[0];
    return (temp<<2);
}

```

RESULT:

a) Waveforms of input and output from CRO



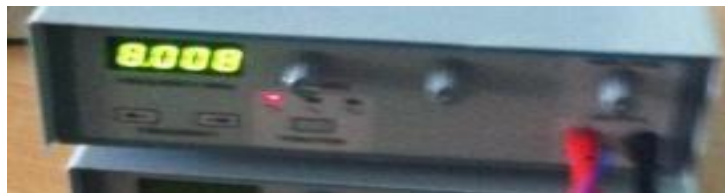
b) function generator (input signal frequency at 1KHz)



b) C6713 DSK



c) function generator (input signal frequency at 8kHz)



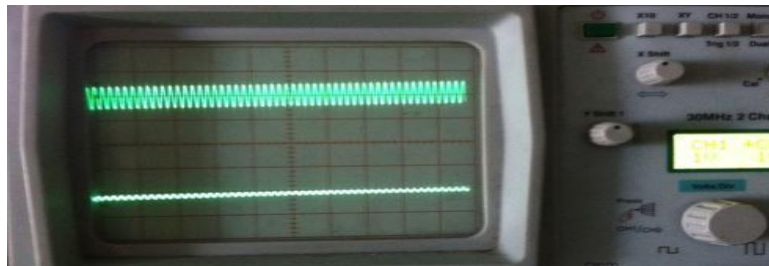
d) Waveforms of input and output from CRO



e) function generator (input signal frequency at 10kHz)



g) Waveforms of input and output from CRO (output signal attenuated)



Insert Graph Sheet t

Annexure –I

VIVA QUESTIONS

GENERATION OF SINUSOIDAL SIGNAL QUESTIONS

1. What is the difference between sin & cos signals?
2. What is meant by signal?
3. What is the difference between time domain & frequency domain signal?
4. What is the difference between periodic & a periodic signal.
5. What is the difference between orthogonal and orthonormal signals?
6. What is the need for Fourier series & Fourier transform?
7. What is the difference between discrete & digital signals?
8. What is the difference between even signal & odd signal?
9. What is the difference between power signal & energy signal?
10. What is the difference between amplitude scaling & time scaling of a signal?
11. What is the difference between deterministic & random signal?

LINEAR CONVOLUTION QUESTIONS

1. What is the requirement for convolution?.
2. What is the difference between convolution & correlation?
3. What is meant by impulse response?
4. Is it possible to represent any discrete time signal in terms of impulses? If yes, represent by using example.
5. Draw the $h(2n-k)$ & $h(n-2k)$ for the following sequence $h(n) = \{ 4 \ 3 \ 2 \ 1 \}$ assume (i) $k= 3$ (ii) $k =5$.
6. Write the expressions for LTI system convolution formula & causal LTI system convolution formula.
7. What us the length of linear convolution if length of input & impulse responses are N_1 & N_2 respectively?
8. What is the difference between continuous and discrete convolution?

CIRCULAR CONVOLUTION QUESTIONS

1. Why we need circular convolution?
2. What is the difference between circular & linear convolution?
3. What is the length of output sequence after circular convolution if the lengths of input & impulse responses are M_1 & M_2 respectively?
4. State the circular convolution property of DFT?
5. Where we required convolution property?
6. What does zero padding mean? Where we required this concept?
7. What is difference between linear shifting & circular shifting of signal? Show with example.
8. What is difference between linear & circular folding of signal? Show with example.
9. What is the advantage with sectioned convolution?

FAST FOURIER TRANSFORM QUESTION

1. What is the difference between continuous time & discrete time Fourier transform?
2. What is the condition for convergence of Fourier transform?
3. What is the difference between discrete Time Fourier Transform (DTFT)& DFT?
4. What is the difference between Z transform & DFT?
5. State convolution property of the DFT? Where we could use the convolution property?
6. State Parseval's theorem.
7. State correlation property of the DFT.?
8. What is the difference between radix 2 & radix4 FFT algorithms?
9. Why we need FFT?.
10. What is the difference between decimation in time (DIT FFT) & Decimation in frequency(DIFFFT) algorithms?
11. What is meant by 'in-place' computation in DIF & DIF algorithms?
12. Which properties are used in FFT to reduce no of computations?

FIR FILTER QUESTIONS

1. What are the advantages of FIR as compared to IIR?
2. How many types of FIR design methods are used in real time?.
3. What is meant by Gibbs Phenomenon? Where we found such type of effect in FIR Filters?

4. What are the advantages & disadvantages of Rectangular window FIR filter as compared to remaining window techniques?
5. Which window technique having less peak amplitude of side lobe as compared to all?
6. What do you understand by linear phase response?
7. To design all types of filters what would be the expected impulse response?
8. What are the properties of FIR filter?
9. How the zeros in FIR filter is located?
10. What are the desirable characteristics of the window?
11. What are the specifications required to design filter

IIR FILTER QUESTIONS

1. What is meant by IIR filter?
2. What is the difference between recursive & non-recursive systems?
3. Write the difference equation for IIR system.
4. What are the mapping techniques in IIR filter design? Discuss the advantage & disadvantages of them.
5. What are IIR analog filters? What are the advantages & disadvantages of them?
6. What is the disadvantage in impulse invariance method?
7. What does warping effect mean? Where we found this effect? How can we eliminate warping effect?
8. Explain the pole mapping procedure of Impulse invariant & bilinear transformation method.
9. For given same specification which difference we found in Butter worth & chebyshev filter.
10. What is the difference between type I & type II chebyshev filters?.
11. Where the poles are located for Butter worth & chedbyshev filters?
12. What is meant by spectral transformation?
13. Why we need spectral transformation in IIR filter?

POWER SPECTRUM DENSITY QUESTION

1. What is the difference between correlation & auto correlation function?
2. What is the difference between PSD & ESD?

3. What is the unit for energy density spectrum?
4. What is the formula for PSD of a function?
5. “Same power density spectrum signals always have same magnitude & phase spectrums”
Is above statement true (or) False: Justify your answer.
6. If we know the impulse response of the system, the How can you find output signal power density from the input signal?
7. What is the unit for power density spectrum?
8. What is the relation between auto correlation & PSD of a function?

DSP PROCESSORS QUESTIONS

1. How many types of DSP processors are available in the market?
2. TMS 320C6X, ‘C’ stands for what?
3. What are the features of TMS 320C6X processor?
4. What is meant by VLIW architecture? Why we required in DSP processor?
5. How many functional units are in TMS 320C6X DSP processor?
6. What is meant by Circular addressing mode how is it useful for DSP?
7. Which instruction is used to move 16 bit constant in to the upper bits of a register?
8. What is the difference between Von Neumann architecture & Harvard architecture?
9. Which architecture is used in DSP processor?
10. How many instructions can we execute per cycle in TMS320C6X DSP processor?
11. What are the applications for the TMS320 DSP’s?
12. Which soft ware tool is required to compile and run the DSP assembly program?
13. What is the difference between full version Code composer studio &DSK CCS?