LAB MANUAL

STUDENT VERSION

for

MICROCONTROLLER LAB

(SIXTH SEMESTER, COURSE CODE: 6139)

Department Of Computer Engineering

GOVERNMENT POLYTECHNIC COLLEGE, PERUMBAVOOR

Koovappady P.O, Phone: 0484 2649251

email:gptcpbvr@gmail.com

This page is intentionally kept blank to preserve page format

INDEX

SL NO:	NAME OF EXPERIMENT	DATE	PAGE NO:
	VISION AND MISSION		
	PEO, PO AND PSOS OF THE PROGRAM		
	GENERAL INSTRUCTIONS		
1	SAFETY PROCEDURES		
2	INSTALLING AVR STUDIO		
3	WORKING WITH AVR STUDIO		
4	ADDITION OF DIFFERENT DATAFORMATS		
5	ARITHMETIC OPERATIONS ON NUMBERS		
6	SHIFT AND ROTATE INSTRUCTIONS		
7	ASCII TO PACKED BCD		
8	PACKED BCD TO ASCII		
9	I/O PORT PROGRAMMING		
10	BIT MANIPULATION-I		
11	TIME DELAY IN ASSEMBLY		
12	BIT MANIPULATION -II		
13	DATA SERIALIZATION IN C		
14	SHIFT AND ROTATE INSTRUCTIONS IN AVR C		
15	ASCII TO PACKED BCD		
16	PACKED BCD TO ASCII		
17	I/O PORT PROGRMMING AVR C		
18	TIME DELAY IN AVR C		
19	TIMER/COUNTER PROGRAMMING		
20	4x4 KEYBOARD INTERFACING		
21	LCD INTERFACING		

GENERAL REMARKS (*FOR OFFICE USE ONLY*)

TEST 1:		TEST 2:		ASSIGN 1:		ASSIGN 2:	

VISION AND MISSION

Government Polytechnic College, Perumbavoor
Vision: Excel as a centre of skill education moulding professionals who sincerely strive for the betterment of society **Mission:** • To impart state of the art knowledge and skill to the graduate and moulding them to be competent, committed and responsible for the well-being of society • To apply technology in the traditional skills, thereby enhancing the living standard of the community

Department of Computer Engineering
Vision: Excel as a center of skill education in Computer Engineering moulding professionals who sincerely strive for the betterment of themselves and the society. **Mission:** • To impart state of the art knowledge, skill and attitude to the graduates and contribute to their sustainable development • To merge technologies in the field of computer engineering with occupational skills, thereby improving the quality of living

PEO, PO AND PSOs OF THE PROGRAM

PROGRAM OUTCOMES

PO1: Basic and Discipline specific knowledge: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.

PO2: Problem analysis: Identify and analyse well-defined engineering problems using codified standard methods.

PO3: Design/ development of solutions: Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

PO4: Engineering Tools, Experimentation and Testing: Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.

PO5: Engineering practices for society, sustainability and environment: Apply appropriate technology in context of society, sustainability, environment and ethical practices.

PO6: Project Management: Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.

PO7: Life-long learning: Ability to analyse individual needs and engage in updating in the context of technological changes.

PROGRAM SPECIFIC OUTCOMES (PSOS)

PSO1: Apply concepts and knowledge in the field of software systems, hardware and networking with concern for the society.

PSO2: Generate ideas from the knowledge of engineering specialization leading to professional growth.

PSO3: Apply knowledge and understanding of engineering principles to initiate entrepreneurship ventures.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO1: Secure successful careers in hardware and software design, development, testing, maintenance and marketing.

PEO2: Acquire knowledge and competency in the domain to develop innovative, cost effective and socially acceptable solutions to engineering problems in a multi-disciplinary work environment.

PEO3: Develop strong fundamental knowledge that prepares them for professional careers/ higher studies with attitude for lifelong learning.

PEO4: Instill the attitude to be sensitive to ethical, societal and environmental issues while pursuing their professional duties.

PEO5: Possess leadership qualities and be effective communicator to work efficiently with diverse teams, promote and practice appropriate ethical practices.

GENERAL INSTRUCTIONS

Rough record and Fair record are needed to record the experiments conducted in the laboratory. Rough records are needed to be certified immediately on completion of the experiment. Fair records are due at the beginning of the next lab period. Fair records must be submitted as neat, legible, and complete.

INSTRUCTIONS TO STUDENTS FOR WRITING THE FAIR RECORD

In the fair record, the index page should be filled properly by writing the corresponding experiment number, experiment name , date on which it was done and the page number.

On the right side page of the record following has to be written:

1. Title: The title of the experiment should be written in the page in capital letters.

2.Exp No: And Date: In the top margin, experiment number and date should be written.

3. Aim: The purpose of the experiment should be written clearly.

4.Principle/Theory: Simple algorithm should be written

5. Procedure: Steps for doing the experiment.

6. Program: Simple working of the algorithm should be written.

7. Results: The results of the experiment must be summarized in writing and should be fulfilling the aim.

On the Left side page of the record following has to be recorded:

1. **Input:** Input of the program given
2. **Output :** Output of the program
3. **Design:** The design of the output (if necessary).

Exp No : 01 **Date :** D D / M M / Y Y

SAFETY PROCEDURES

Problem Statement:

The safety instructions are presented to the attention of the students as a mean of preventing accidents while performing experiments and activities in Software lab of the department .The purpose is to draw attention to the risks involved in lab activities to prevent human suffering and damage to equipment.

Safety in the laboratory:

Working in the lab is not allowed without following electricity precautions displayed.
No individual work is allowed in the lab.
Laboratory in charge is responsible for the arrangements of your lab activities; Listen carefully to his/her instructions and follow them.

To do and not to do:

Inform the lab in charge about dangerous conditions and faults in the lab or nearby environment.
Do not do any action that may harm people or equipment in the lab.
Do not misuse any of the tools or instruments belong to the lab.
Strict discipline should be maintained in the laboratory.
Turn off cell phones before entering the lab.
At the end and beginning of laboratory, follow 5S procedures and leave the work table clean and tidy.

Electrical Safety:
Consult Electrical Engineering section available in the campus for electrical safety queries.
The lab equipment is powered from electrical sockets installed on the tables.
Do not use equipment that is powered from a damaged socket.
Do not use equipment that is powered from flexible cable with damaged insulation or if it's plug is not assembled properly.

Do not repair or disassemble electrical equipment including replacement of fuses installed in the equipment.
Do not open the main fuse box, unless it is an emergency and you need to switch off main circuit breaker.
Be sure to turn off the power and remove the power plug from all equipment before working repairing or assembling.
Do not plug in or remove equipment while the power is on.

Emergency Switches:
The laboratory has circuit breakers, which is located in the main panel. Identify the place.
In an emergency condition, switch off circuit breakers immediately.

Result:
Familiarization of safety precautions performed

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 02 **Date :** D D / M M / Y Y

INSTALLING AVR STUDIO

AIM:

To Familiarize Installing AVR Studio.

PROCEDURE:

Any microcontroller requires software called Integrated Development Environment – IDE for writing program. An IDE is the handy software that acts as text editor, debugger, assembler and compiler all in one package. A text editor is simply like note pad software for writing the code. The text editor comes as a package along with the IDE. Here in this book whatever program we write for microcontroller will be called as source code or simply code. An assembler interprets a code written in assembly language to machine code. The compiler converts codes written in C into machine codes. In our case, the compiler is called cross compiler. A cross compiler converts instructions into machine code or low-level code for a computer other than that on which it is run. Here the target computer is the ATMEGA micro controller. A debugger assists detection and correction of errors in code.

Atmel AVRStudio is the Integrated Development Environment (IDE) for developing and debugging embedded Atmel AVR applications. The AVR Studio IDE gives you a seamless and easy-to-use environment to write, build, and debug your C/C++ and assembler code. AVR studio can be down loaded from www.atmel.com.

Three popular versions of AVR studio are available for use with ATMEGA processors. AVR studio 7 is latest addition into the series.

- AVR Studio Ver.4
- AVR Studio Ver.5
- AVR Studio Ver.6

All you have to do is to down load the version that is most suitable for your computer operating system. For example, for windows XP service pack 2 you need to install AVR Studio 4. IF you encounter with any error while installing the Studio, try for the other versions.

AVR studio for Linux environment can also be down loaded. If you are installing AVR Studio 4, you have to install Win AVR - the compiler separately. Win AVR

should be installed only after installation of AVR studio 4. But for all other versions, the compiler comes along with the package. Go for the latest version if your operating system supports.

Installations of this software are pretty straightforward. You should not face any trouble in it. During the installation of AVR Studio 5 or 6, it might need to install some other stuff before the actual installation begins. These installations will be initiated by itself.

RESULT:

Familiarized the installation of AVR Studio.

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 03 **Date :** D D / M M / Y Y

WORKING WITH AVR STUDIO

AIM:

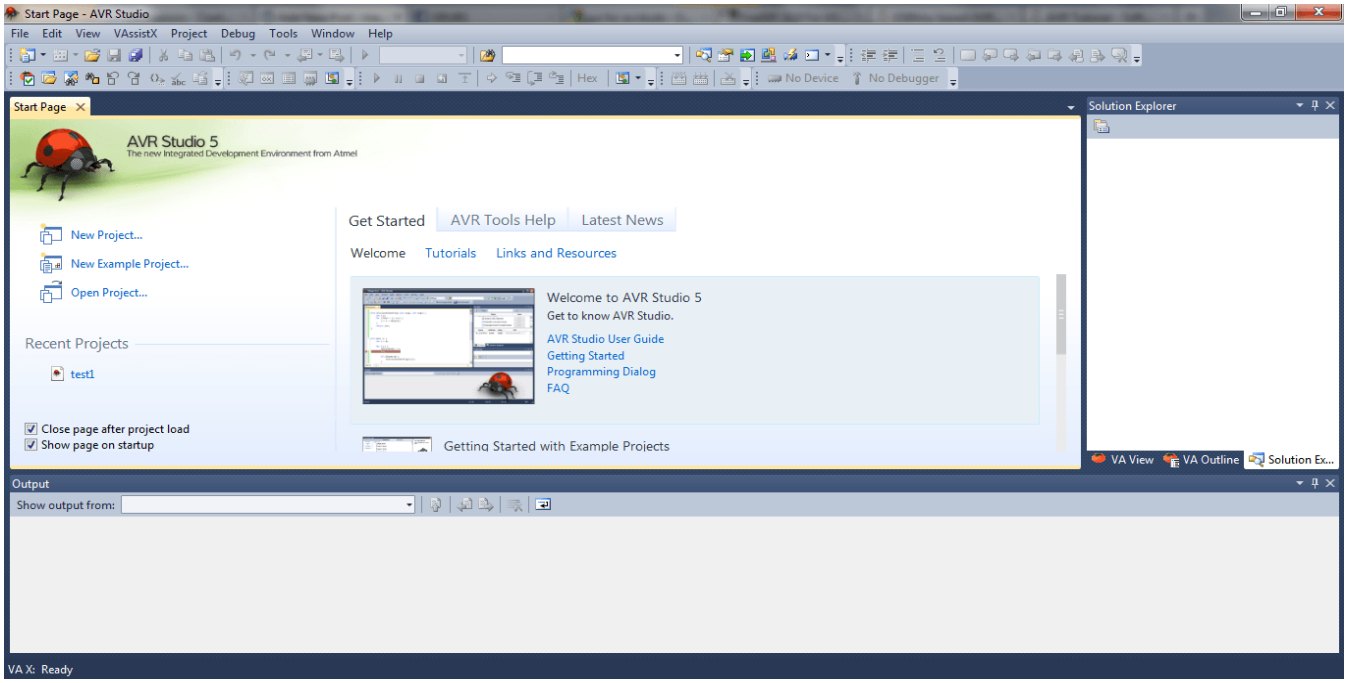
To Familiarize the Working of AVR Studio.

Features of AVR Studio:

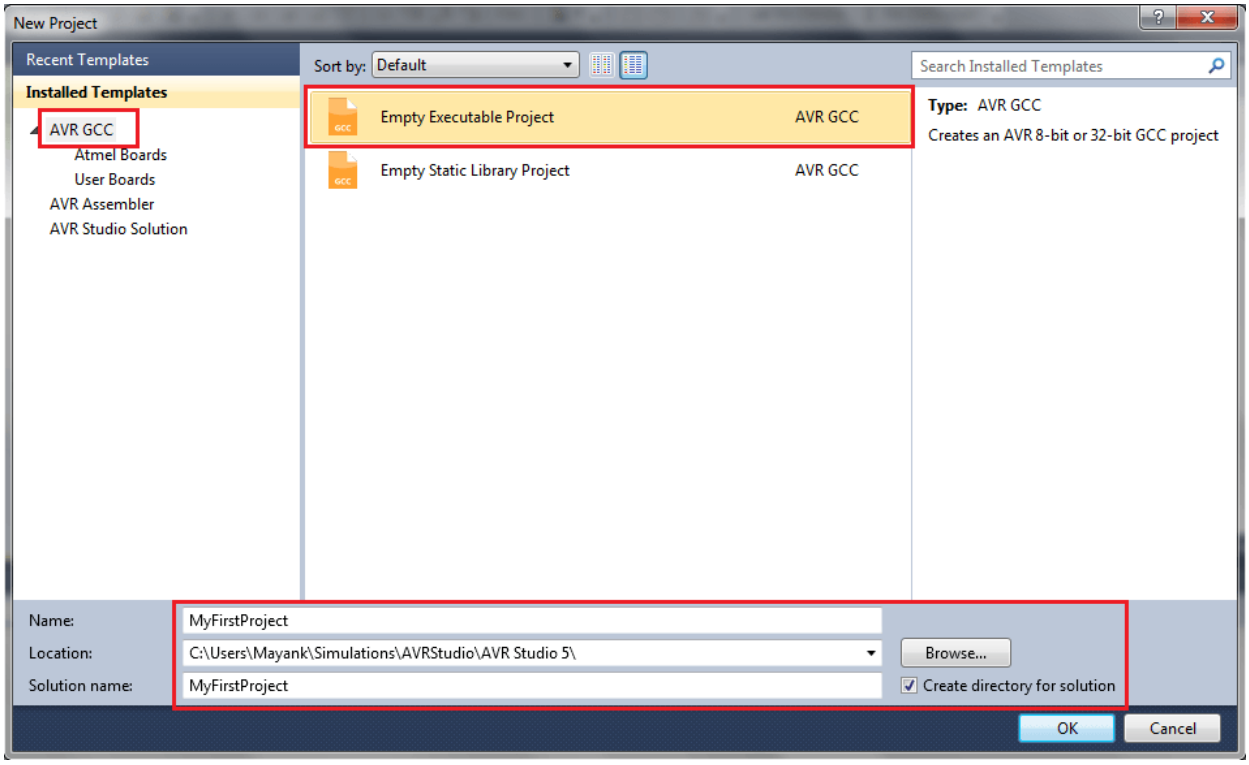
- It is an Integrated Development Environment (IDE) for AVR Software.
- It allows chip simulation and in-circuit emulation.
- It supports the whole AVR family of Microcontrollers (MCUs).
- It has easy to use User Interface (UI) and gives complete overview.
- It uses same UI for simulation and emulation.

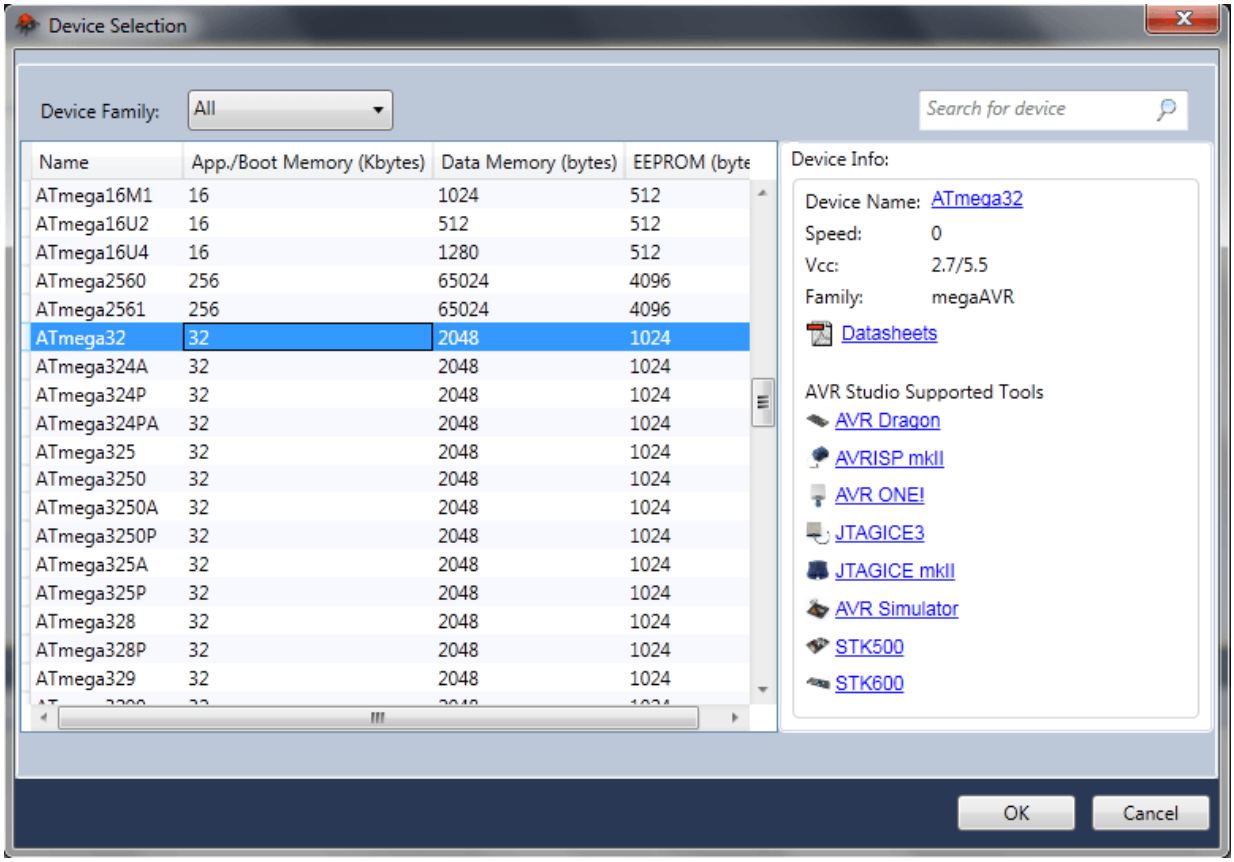
PROCEDURE:

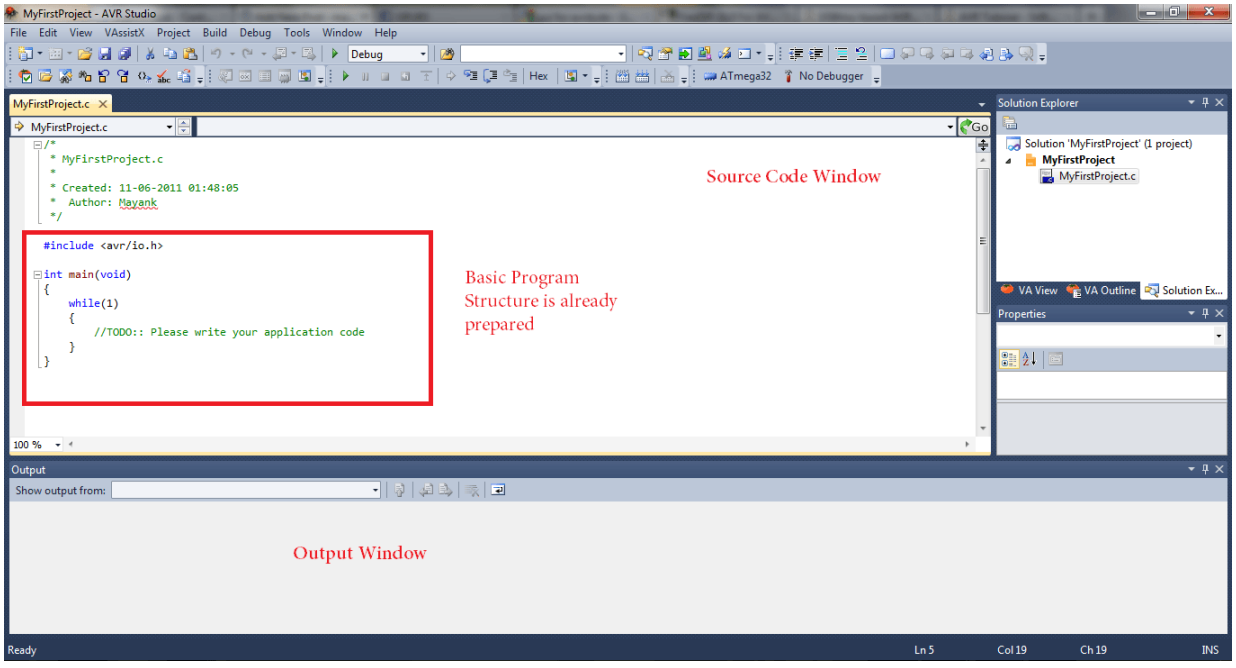
- Creating First AVR Project:



- After installation, open AVR Studio 5 from Start → All Programs → Atmel AVR Tools → AVR Studio 5.0
- After opening, you will see a Start Page like this. Click on "New Project…"
- Then, you will see the following dialog box. Choose AVR GCC from the 'Installed Templates' pane, and then choose Empty Executable Project. Now, you can give any name to it, say MyFirstProject and choose an appropriate location in your hard drive. Check Create directory for solution. Click on OK.
- Now, you will see the Device Selection dialog box. As you can see, AVR Studio supports all the AVR MCUs! The list is huge! Choose your device from this list. We choose ATmega32. Click OK.




Device Selection
Device Family:
All
Search for device
Name
App./Boot Memory (Kbytes)
Data Memory (bytes)
EEPROM (byte
ATmega16M1 16 1024 512
ATmega16U2 16 512 512
ATmega16U4 16 1280 512
ATmega2560 256 65024 4096
ATmega2561 256 65024 4096
ATmega32 32 2048 1024
ATmega324A 32 2048 1024
ATmega324P 32 2048 1024
ATmega324PA 32 2048 1024
ATmega325 32 2048 1024
ATmega3250 32 2048 1024
ATmega3250A 32 2048 1024
ATmega3250P 32 2048 1024
ATmega325A 32 2048 1024
ATmega325P 32 2048 1024
ATmega328 32 2048 1024
ATmega328P 32 2048 1024
ATmega329 32 2048 1024
Device Info:
Device Name: ATmega32
Speed: 0
Vcc: 2.7/5.5
Family: megaAVR
Datasheets
AVR Studio Supported Tools
AVR Dragon
AVRISP mkII
AVR ONE!
JTAGICE3
JTAGICE mkII
AVR Simulator
STK500
STK600
OK
Cancel


MyFirstProject - AVR Studio
File Edit View VAssistX Project Build Debug Tools Window Help
MyFirstProject.c
/*
* MyFirstProject.c
*
* Created: 11-06-2011 01:48:05
* Author: Mayank
*/
#include <avr/io.h>
int main(void)
{
while(1)
{
//TODO:: Please write your application code
}
}
Source Code Window
Basic Program
Structure is already
prepared
Solution Explorer
Solution 'MyFirstProject' (1 project)
MyFirstProject
MyFirstProject.c
Properties
Output
Show output from:
Output Window
Ready
Ln 5
Col 19
Ch 19
INS

- Now, you will see the following screen. Note that the basic program structure is already prepared for you. You simply need to initialize your variables and place your code inside the while(1) loop.

- Now you have successfully created your first AVR Project! Now to further proceed with it, we need to write a code, which is discussed in the subsequent chapters.

RESULT:

Familiarized the Working of AVR Studio.

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 04 **Date :** D D / M M / Y Y

ADDITION OF DIFFERENT DATAFORMATS

AIM:

Write an AVR assembly language program to perform the Addition of numbers various data formats.

OBJECTIVE

After completing this experiment the student will able to understand various data formats of AVR microcontroller. THEORY AVR microcontroller supports various data formats like binary, hexadecimal and decimal.

- Binary - A binary number is a combination of 0's and 1's. E.g: LDI R16,0B00101101
- Decimal - A decimal number system consists of numbers between 0-9. E.g: LDI R17,10
- Hexa Decimal - An hexadecimal number system consist of numbers 0-9 and Alphabets A-F. E.g: LDI R18,22H
- ASCII - ASCII means American standard code for information interchange Is a character encoding standard for electronics communication. ASCII (American Standard Code for Information Interchange) is the most common format for text files in computers and on the Internet. In an ASCII file, each alphabetic, numeric, or special character is represented with a 7- bit binary number (a string of seven 0s or 1s). 128 possible characters are defined. E.g:LDI R16,'32'

SOURCE CODE

SAMPLE OUTPUT:

INPUT VALUES:

OUTPUT
AFTER 1ST ADDITION ----------→
AFTER 2ND ADDITION----------→

INPUT VALUE

OUTPUT
AFTER FINAL ADDITION---------→

RESULT

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 05 **Date :** D D / M M / Y Y

ARITHMETIC OPERATIONS ON NUMBERS

AIM

Write an AVR assembly language program to perform various arithmetic operations.

OBJECTIVE

After completing this experiment the student will able to understand various arithmetic operations in AVR microcontroller, and a generalized concept of arithmetic operators in AVR.

THEORY

AVR microcontroller supports various arithmetic operations like addition, subtraction, multiplication etc.some of the important arithmetic operators are given below:

- ADD Rd, Rr -It adds the contents of the two registers Rd,Rr and places The result in the Destination register Rd.
  Eg: ADD R16,R17
- SUB Rd, Rr - It subtracts the values of the two registers Rd,Rr and Places the result in the Destinations register Rd.
  Eg:SUB R17,R16
- MUL Rd, Rr -It multiplies the values of the two registers Rd,Rr andPlaces the result in the Destination register Rd.
  Eg: MUL R17,R16
- SUBI Rd, K - It subtracts a register and a constant and places the resultIn the destination register Rd.
  Eg:SUBI R16,3
- ADI Rd, K -It adds a register and a constant and places the result in the Destination register Rd.
  Eg: ADI R16,3

SOURCE CODE

SAMPLE OUTPUT

INPUT:

OUTPUT:

R16= (ADDITION RESULT)
R19= (SUBTRACTION RESULT)
R18= (MULTIPLICATION RESULT)

RESULT

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 06 **Date :** D D / M M / Y Y

SHIFT AND ROTATE INSTRUCTIONS

AIM

Write an AVR Assembly language program to count number of 0's and 1's in a binary number.

OBJECTIVE

After completion of this experiment the student able to understand the usage of shift and rotation instructions.

THEORY

Shift and Rotate instructions shifts the bits in the destination operand by one or more position either right or left.

1. Various Shift instructions :
   - Shift Left
     - Instruction Format:
       LSL Destination, bits_shifted
     - The Shift Left instruction performs a left shift on the destinations operand, filling the lowest bit with 0. The highest bit is moved into the Carry Flag.
     - Eg: MOV CL,5
       LSL AX,CL
   - Shift Right
     - Instruction format:
       LSR destination, bits shifted
     - The Shift Right instruction performs a right shift on the destinations operand, filling the lowest bit with 0. The lowest bit is moved into the Carry Flag.
     - Eg: MOV CL,5
       LSR AX,CL
   - Arithmetic Shift Right
     - Shift all bits in Rd one places to the right. The least significant bit is discarded and the MSB value is filled with the value of the previous MSB.
2. Various Rotate instructions:

- Rotate Left
  - Instruction format:
    ROL destination, bits shifted
  - The ROL instruction shifts each bit to the left, with the highest bit copied in the Carry flag and into the lowest bit.
  - Eg:MOV AL,42H
    ROL AL,1
- Rotate Right
  - Instruction format:
    ROR destination, bits shifted
  - The ROR instruction shifts each bit to the right, with the lowest bit copied in the Carry flag and into the highest bit.
  - Eg: MOV AL,23H
    ROR AL,1

SOURCE CODE

SAMPLE OUTPUT

INPUT :

R16=
R17=
R18=
R19=

OUTPUT:

R17= // NUMBER OF ZEROES
R18= // NUMBER OF ONES

RESULT

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 07 **Date :** D D / M M / Y Y

ASCII TO PACKED BCD

AIM

Write an AVR assembly language program to convert the given ASCII number into Packet BCD.

OBJECTIVE

After completion of this experiment the student will be able to understand the concept simple code conversion

THEORY

To convert ASCII to packed BCD, you first convert it to unpacked BCD, and then combine it to make packed BCD. For example, for 4 and 7 the keyboard gives 34 and 37, respectively. The goal is to produce 4711 or "0100 0111", which is packed BCD. This process is illustrated next.

Key	ASCII	Unpacked BCD	Packed BCD
4	34	00000100	
7	37	00000111	01000111 which is 47H

SOURCE CODE

SAMPLE OUTPUT

INPUT:
R16=
R17=
OUTPUT:
R16=

RESULT

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 08 Date : D D / M M / Y Y

PACKED BCD TO ASCII

AIM

Write an AVR assembly language program to convert the given Packed BCD number into ASCII.

OBJECTIVE

After completion of this experiment the student will be able to understand the concept simple code conversion

THEORY

In many systems we have what is called a *real-time clock* (RTC). The RTC provides the time of day (hour, minute, second) and the date (year, month, day) continuously, regardless of whether the power is on or off (see Chapter 16). This data, however, is provided in packed BCD. For this data to be displayed on a device such as an LCD, or to be printed by the printer, it must be in ASCII format. To convert packed BCD to ASCII, you must first convert it to unpacked BCD. Then the unpacked BCD is tagged with 011 0000 (30H). The following demonstrates converting packed BCD to ASCII.

Packed BCD	Unpacked BCD	ASCII
29H 0010 1001	02H & 09H 0000 0010 & 0000 1001	32H & 39H 0011 0010 & 0011 1001

SOURCE CODE

SAMPLE OUTPUT

INPUT:
R16=
R17=

OUTPUT:
R16=
R17=

RESULT

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 09 **Date :** D D / M M / Y Y

I/O PORT PROGRAMMING

AIM

Write a program to clear R20 register, then add 3 to R20 10 times; then send the sum to PORTB.

OBJECTIVE

After completion of this experiment the student should be able to understand the operation of ATMEGA PORT as output. The operations of PORT are visualized by interfacing LEDs to PORT pins. In addition, this experiment will provide preliminary experience on programming microcontrollers on Embedded C.

THEORY

ATMEGA ports are 8 bit wide. Each port has 3 eight bit registers associated. Each bit in these registers configures pins of associated port. Bit 0 of these registers is associated with Pin 0 of the port, Bit1 of these registers is associated with Pin1 and so on.

These three registers are

- DDRx register
- PORTx register
- PINx register

X may be replaced by A,B,C or D based on the PORT you are using.

❖ DDRx register

DDRx (Data Direction Register) configures data direction of the port pins. Which, writing 0 to a bit in DDRx makes corresponding port pin as input, while writing 1 to a bit in DDRx makes the corresponding port pin as output.

Example:

- to make all pins of port B as input,

DDRA = 0b00000000;

- to make all pins of port A as output pins :

DDRB= 0b11111111;

- to make lower nibble of port B as output and higher nibble as input :

DDRB = 0b00001111; In hexadecimal representation, it can be written as DDRB = 0x0F;

SOURCE CODE

SAMPLE OUTPUT

INPUT:
R16=
R18=
R20=

OUTPUT:
R20=

RESULT

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 10 **Date :** D D / M M / Y Y

BIT MANIPULATION-I

AIM

A switch is connected to pin PB2. Write a program to check the status of the switch and perform followings:

1. If Switch=0, send the letter 'N' to PORTD.
2. If Switch=1, send the letter 'Y' to PORTD.

OBJECTIVE

After completing this experiment the student able to understand the various bit manipulation operators in AVR microcontroller.

THEORY

AVR microcontroller has functionality for single bit manipulation. The various single bit manipulators are:

- ❖ SBI I/O Reg, Bit - Sets a Specified bit in I/O register. This Instruction operates on the lower 32 I/O Registers –addresses 0-31.
  Eg: SBI PORTB,5
- ❖ CBI I/O Reg, Bit - Clears a specified bit in I/O register. This Instruction operates on the lower 32 I/O Register –addresses 0-31.
  Eg: CBI PORTB,5
- ❖ SBIS I/O Reg, Bit - This instruction tests a single bit in an I/O Register and skips the next instruction if the Bit is set. This instruction operates on the Lower 32 I/O register- addresses 0-31.
  Eg: SBIS PORTB,5
- ❖ SBIC I/O Reg, Bit - This instruction tests a single bit in an I/O Register and skips the next instruction if the Bit is cleared. This instruction operates on the Lower 32 I/O register-addresses 0-31.
  Eg: SBIC PORTB,5

SOURCE CODE

SAMPLE OUTPUT

INPUT:
PORTD= //INITIAL VALUE OF PORTD

OUTPUT:
PORTD= // WHEN PB2=0(OUTPUT IS 'N')
PORTD= // WHEN PB2=1(OUTPUT IS 'Y')

RESULT

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 11 **Date :** D D / M M / Y Y

TIME DELAY IN ASSEMBLY

AIM

Write an assembly language program to toggle the bits of PORTD one after another continuously.

OBJECTIVE

After completion of this experiment the student should be able to understand the operation of ATMEGA PORT as output. The operations of PORT are visualized by interfacing LEDs to PORT pins. In addition, this experiment will provide preliminary experience on programming microcontrollers on Embedded C.

THEORY

In creating a time delay using assembly language instructions, one must be mindful of two factors that can affect the accuracy of delay:

- ❖ The Crystal Frequency
- ❖ The AVR Design

➢ Delay calculation in AVR with crystal frequency is 1MHz.

➢

		INSTRUCTION CYCLES
• DELAY:	LDI R20,0XFF	1
AGAIN:	NOP	1
	NOP	1
	DEC R20	1
	BRNE AGAIN	2/1
	RET	4

Therefore, we have a time delay of [1+ (255 *5)-1+4]*1=1279

Here the branch instruction BRNE can have two values either 2 or 1. If the branching condition is true then it will branch therefore it uses two machine cycle otherwise it uses only one machine cycle.

When a branch instruction is executed, the CPU starts to fetch codes from the new memory location, and the code in the queue that was fetched previously is discarded. In this case, the execution unit must wait until the fetch unit fetches the new instruction. This is called branch penalty.

SOURCE CODE

SAMPLE OUTPUT

INPUT:
PORTD =

OUTPUT:
PORTD = // AFTER FIRST EXECUTION
PORTD = // AFTER SECOND EXECUTION

RESULT

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 12 **Date :** D D / M M / Y Y

BIT MANIPULATION -II

AIM

Write an assembly language program to set and clear the third pin (PB2) of PINB continuously.

OBJECTIVE

After completing this experiment the student able to understand the various bit manipulation operators in AVR microcontroller.

THEORY

AVR microcontroller has functionality for single bit manipulation. The various single bit manipulators are:

- SBI I/O Reg, Bit - Sets a Specified bit in I/O register. This Instruction operates on the lower 32 I/O Registers –addresses 0-31. Eg: SBI PORTB,5
- CBI I/O Reg, Bit - Clears a specified bit in I/O register. This Instruction operates on the lower 32 I/O Register –addresses 0-31. Eg: CBI PORTB,5
- SBIS I/O Reg, Bit - This instruction tests a single bit in an I/O Register and skips the next instruction if the Bit is set. This instruction operates on the Lower 32 I/O register- addresses 0-31. Eg: SBIS PORTB,5
- SBIC I/O Reg, Bit - This instruction tests a single bit in an I/O Register and skips the next instruction if the Bit is cleared. This instruction operates on the Lower 32 I/O register-addresses 0-31. Eg: SBIC PORTB,5

SOURCE CODE

SAMPLE OUTPUT

INPUT:
PORTB=

OUTPUT:
PORTB= //AFTER FIRST EXECUTION
PORTB= //AFTER SECOND EXECUTION
PORTB= //AFTER THIRD EXECUTION

RESULT

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

AVR C PROGRAMS

Exp No : 13 **Date :** D D / M M / Y Y

DATA SERIALIZATION IN C

AIM

Write an AVR C program to send out the value 44H serially one bit at a time via PORTC, The LSB should go out first.

OBJECTIVE

After completing this experiment the student will able to understand the data serialization in AVR microcontroller, and a generalized concept of various bitwise operators in AVR C.

THEORY

Serializing data is a way of sending a byte of data one bit at a time through a single pin of the microcontroller. There are two ways to transfer a byte of data serially:

1. Using the Serial Port.

2. The second method of serializing data is to transfer data one bit at a time and control the sequence of data and spaces between them.

3. The data serialization can be done in either bit wise or byte wise

• We can send the data bit-wise to a particular pin and also we can store a byte of data in continuous bits of an i/o port.

- The AVR C supports various bitwise operators such as,

❖ Shift Right It shifts the bits to right by a specified number of time.

Format: data >> number of bits to be shifted right.

E.g: 0b00010000 >>3 =0b00000010

❖ Shift left It shifts the bits to left by a specified number of time.

Format: data << number of bits to be shifted left.

E.g: 0b00010000 <<3 = 0b10000000

SOURCE CODE

OUTPUT

INPUT: Y =

OUTPUT: PORTC (PIN3)= //AFTER 1ST LOOP EXECUTION
PORTC(PIN3)= // AFTER 3RD LOOP EXECUTION

RESULT

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 14 **Date :** D D / M M / Y Y

SHIFT AND ROTATE INSTRUCTIONS IN AVR C

AIM

Write an AVR C program to count the number of 1's and 0's in a given binary number.

OBJECTIVE

After completion of this experiment the student able to understand the usage of various shift and rotation instructions.

THEORY

Shift and Rotate instructions shifts the bits in the destination operand by one or more position either right or left.

1. Various Shift instructions
   - ➢ Shift Left
   - • Instruction Format: Data << Number of bits to be shifted left.
   - •This instruction shifts the data to a left by a specified number of times
   - • Eg: 0B00000010 << 3 = 0B00010000
   - ➢ Shift Right
   - • Instruction format: Data >> Number of bits to be shifted right.
   - • This instruction shifts the data to right by a specified number of times.
   - • E.g : 0B00010000 >> 3 = 0B00000010

SOURCE CODE

OUTPUT

INPUT: y =
OUTPUT:
PORT B= //NUMBER OF ZEROES
PORT C= //NUMBER OF ONES

RESULT

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 15 **Date :** D D / M M / Y Y

ASCII TO PACKED BCD

AIM

Write an AVR C program to convert the given ASCII number into Packet BCD.

OBJECTIVE

After completion of this experiment the student will be able to understand the concept simple code conversion

THEORY

To convert ASCII to packed BCD, you first convert it to unpacked BCD, and then combine it to make packed BCD. For example, for 4 and 7 the keyboard gives 34 and 37, respectively. The goal is to produce 4711 or "0100 0111", which is packed BCD. This process is illustrated next.

Key	ASCII	Unpacked BCD	Packed BCD
4	34	00000100	
7	37	00000111	01000111 which is 47H

SOURCE CODE

OUTPUT

INPUT:
x= ’ ’
y= ’ ’
OUTPUT:

PORTB=

RESULT

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 16 **Date :** D D / M M / Y Y

PACKED BCD TO ASCII

AIM

Write an AVR C program to convert the given Packed BCD number into ASCII.

OBJECTIVE

After completion of this experiment the student will be able to understand the concept simple code conversion

THEORY

In many systems we have what is called a real-time clock (RTC). The RTC provides the time of day (hour, minute, second) and the date (year, month, day) continuously, regardless of whether the power is on or off (see Chapter 16). This data, however, is provided in packed BCD. For this data to be displayed on a device such as an LCD, or to be printed by the printer, it must be in ASCII format. To convert packed BCD to ASCII, you must first convert it to unpacked BCD. Then the unpacked BCD is tagged with 011 0000 (30H).

The following demonstrates converting packed BCD to ASCII.

Packed BCD	Unpacked BCD	ASCII
29H	02H & 09H	32H & 39H
0010 1001	0000 0010 & 0000 1001	0011 0010 & 0011 1001

SOURCE CODE

OUTPUT

INPUT:

x=

OUTPUT:

PORT B=
PORT C=

RESULT

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 17 **Date :** D D / M M / Y Y

I/O PORT PROGRMMING AVR C

AIM

Write an AVR C program to get a data from PINB and send to the I/O register of PORTC continuously.

OBJECTIVE

After completion of this experiment the student should be able to understand the operation of ATMEGA PORT as output. The operations of PORT are visualized by interfacing LEDs to PORT pins. In addition, this experiment will provide preliminary experience on programming microcontrollers on Embedded C.

THEORY

ATMEGA ports are 8 bit wide. Each port has 3 eight bit registers associated. Each bit in these registers configures pins of associated port. Bit 0 of these registers is associated with Pin 0 of the port; Bit1 of these registers is associated with Pin1 and so on.

These three registers are

➢ DDRx register

➢ PORTx register

➢ PINx register X may be replaced by A,B,C or D based on the PORT you are using.

❖ DDRx register DDRx (Data Direction Register) configures data direction of the port pins. Which, writing 0 to a bit in DDRx makes corresponding port pin as input, while writing 1 to a bit in DDRx makes the corresponding port pin as output.

Example:

• to make all pins of port B as input, DDRA = 0b00000000;

• to make all pins of port A as output pins : DDRB= 0b11111111;

• to make lower nibble of port B as output and higher nibble as input : DDRB = 0b00001111;

In hexadecimal representation, it can be written as DDRB = 0x0F;

SOURCE CODE

SAMPLE OUTPUT

INPUT:
PINB=

OUTPUT:

PORTC=

RESULT

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 18 **Date :** D D / M M / Y Y

TIME DELAY IN AVR C

AIM

Write an AVR C program to toggle bits 1 and 3 of PORTB in every 200ms.

OBJECTIVE

After completion of this experiment the student should be able to understand the operation of ATMEGA PORT as output. The operations of PORT are visualized by interfacing LEDs to PORT pins. In addition, this experiment will provide preliminary experience on programming microcontrollers on Embedded C.

THEORY

There are three ways to create a time delay in AVR C :

- Using a simple for loop
  - Factors affecting the accuracy of delay:
    - The Crystal Frequency connected to the XTAL1- XTAL2 input pins.
    - Compiler used to compile the C program
    -
- Using pre defined C functions
  - _delay_ms()
  - _delay_us()
  -
- Using AVR Timers.

SOURCE CODE

OUTPUT

1ST LOOP EXECUTION RESULT

----------→PB1=

----------→PB3=

2ND LOOP EXECUTION RESULT

----------→PB1=

----------→PB3=

RESULT

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 19 **Date :** D D / M M / Y Y

TIMER/COUNTER PROGRAMMING

AIM

Write an AVR C program to toggle all bits of PORTB alternatively, with some delay. Use TIMER0 normal mode and pre scalar option to generate delay.

OBJECTIVE

After completing this experiment the student will able to understand the Usage of timers in AVR microcontroller.

THEORY

A Timer is a simple counter. The input clock of microcontroller and operation of the timer is independent of the program execution. The Timers are mainly classified into three:

❖ TIMER 0

• It is an 8-bit timer.
• The register where the counting takes place is the TCNTn register, where n is become 0, 1, 2 etc. It counts automatically and overflows and restarts again.
• CS02, CS00 bits in the TCCR0 are used to choose the clock source.

The various modes of clock source are:

D2	D1	D0	
0	0	0	- Timer/counter Stops.
0	0	1	- No Pre scalar
0	1	0	- Clk / 8
0	1	1	- Clk / 64
1	0	0	- Clk /256
1	0	1	- Clk /1024
1	1	0	- External source to falling edge.
1	1	1	- External Source to raising edge.

• WGM is used for Timer mode selection.
Various modes are:

D6	D3	
0	0	- Normal mode
0	1	- CTC (Clear Timer on Compare Match)
1	0	- PWM, phase correct
1	1	- Fast PWM

• TIFR(Timer/counter flag register) – contains flags of different timers. Important flags are:

1. TOV0 D0 –
   Timer0 overflow flag bit to 0 – Timer0 did not overflow.
   Timer0 overflow flag bit to 1 – Timer0 has overflowed.
2. OCF0 D1 –
   Timer0 output compare flag to 0 – Compare match did not occur.
   Timer0 output compare flag to 1 – Compare match occurred.

❖ TIMER 1

• It is a 16-bit timer.
• The control register split into 2 8-bit registers TCCR1A and TCCR1B.

❖ TIMER 2

• It is an 8-bit timer.
• It cannot be used as a counter, because it does not support external clock.
• It can be used as a real time clock.

SOURCE CODE

OUTPUT

BEFORE EXECUTION----------→: PORTB=
AFTER DELAY----------→:PORTB=

RESULT

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 20 Date : D D / M M / Y Y

4x4 KEYBOARD INTERFACING

AIM

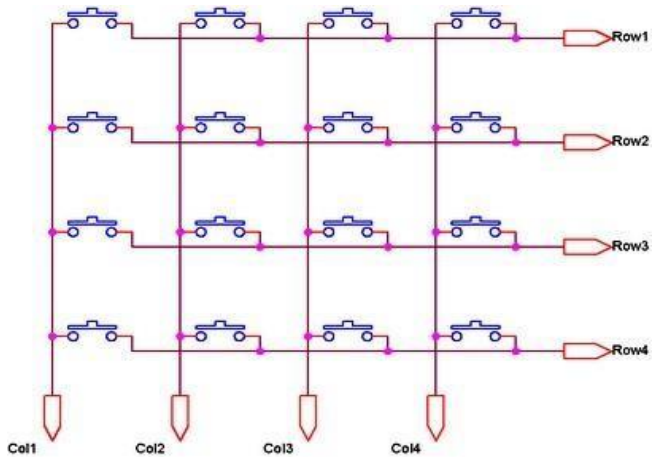
To interface 4x4 matrix keyboard PORTD of ATMEGA32 and to output key
press values in ASCII to PORTB.

OBJECTIVE

After completion of this experiment the student will be able to interface 4x4 keyboards. He / She will be able to generalize embedded C programs.

THEORY

Matrix Keypads are commonly used in calculators, telephones etc where a large number of input switches are required. We know that matrix keypad is made by arranging push button switches in row and columns. If the 16 switches are straightly connected to microcontroller we need 16 inputs pins. But arranging switches in matrix form, we can read the status of each switch using 8 pins of the microcontroller.

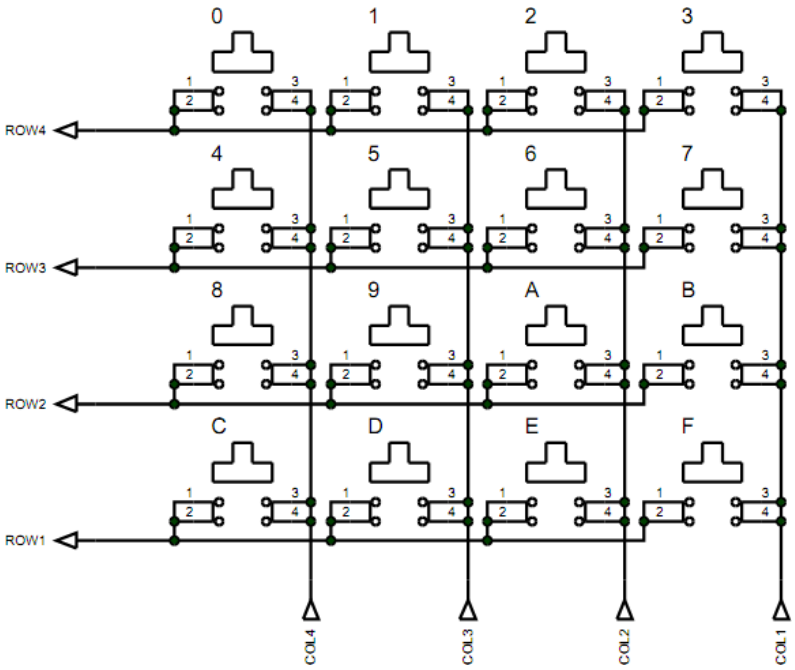


The status of each key can be determined by a process called Scanning. For the sake of explanation let's assume that all the column pins (Col1 – Col4) are connected to the inputs pins and all the row pins are connected to the output pins of the microcontroller. In the normal case all the column pins are pulled up (HIGH state) by internal pull up resistors. Now we can read the status of each switch through scanning.

1. A logic LOW is given to Row1 and others (Row2 – Row-4) HIGH

2. Now each Column is scanned. If any switch belongs to 1st row is pressed corresponding column will pulled down (logic LOW) and we can detect the pressed key.

3. This process is repeated for all rows.



PROGRAM

```
/*---------------------------------------------------------- Program to interface 4x4 matrix keyboard. rows are connected to PD4 to PD7 columns to PD0 to PD3 and display the contents IN ASCII to LEDs connected to PORTB ----------------------------------------------------------------*/

#define

#include

#include
/*the following is done as the part of generalizing the code This code can be used any where, all you need to do is change PORT below*/

#define

#define

#define

//---------------------- MAIN FUNCTION ------------------------//
```

Program

OUTPUT

RESULT

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		

Exp No : 21 **Date :** D D / M M / Y Y

LCD INTERFACING

AIM

To interface 16x2 alphanumeric LCD display in 8 bit mode to display "IF YOU CAN READ THIS, PROGRAM OK". Use PORTB for D0 to D7, RS – PD4, RW – PD5 and EN – PD6.

OBJECTIVE

After completion of this experiment the student will be able to interface LCD displays to micro controller.

He / She will be able to display any required string.

THEORY

LCD display is an inevitable part in almost all embedded projects. We will look about interfacing 16×2 LCD with ATMEGA. In order to understand the interfacing first you have to know about the module. It consists of 16 rows and 2 columns of 5×7 or 5×8 LCD dot matrices. They are available in a 16 pin package with back light, contrast adjustment function and each dot matrix has 5×8 dot resolution. The pin numbers, their name and corresponding functions are shown in the table below.

Pin No:	Name	Function
1	VSS	This pin must be connected to the ground
2	VCC	Positive supply voltage pin (5v DC)
3	VEE	Contrast Adjustment
4	RS	Register Selection
5	R/W	Read or Write
6	E	Enable
7	DB0	Data
8	DB1	Data

9	DB2	Data
10	DB3	Data
11	DB4	Data
12	DB5	Data
13	DB6	Data
14	DB7	Data
15	LED+	Back Light LED +
16	LED-	Back Light LED -

LCD COMMANDS

Command	Function
0F	LCD ON, Cursor ON, Cursor blinking ON
01	Clear Screen
02	Return Home
04	Decrement Cursor
06	Increment Cursor
0E	Display ON, Cursor blinking OFF
80	Force Cursor to beginning of 1st line
C0	Force Cursor to beginning of 2nd line
38	Use 2 lines and 5x7 Matrix
83	Cursor line 1 Position 3
3C	Activate 2nd line
08	Display OFF, Cursor OFF
C1	Jump to 2nd line Position 1
0C	Display ON, Cursor OFF
C1	Jump to second line, position one
C2	Jump to second line , position two

LCD INITIALIZATION

The steps that have to be done for initializing the LCD display is given below and these steps are common for almost all applications.

- Send 38H to the 8 bit data line for initialization
- Send 0FH for making LCD ON, cursor ON and cursor blinking ON.

- Send 06H for incrementing cursor position.
- Send 01H for clearing the display and return the cursor.

SENDING DATA TO THE LCD

The steps for sending data to the LCD module are given below. I have already said that the LCD module has pins namely RS, R/W and E. It is the logic state of these pins that make the module to determine whether a given data input is a command or data to be displayed.

- Make R/W low.
- Make RS=0 if data byte is a command and make RS=1 if the data byte is a

data to be displayed.

- Place data byte on the data register.
- Pulse E from high to low.
- Repeat above steps for sending another data.

PROGRAM

```
/* ---------------------------------------------------- This program displays string given. It
uses RS - PD4 RW - PD5 AND EN - PD6 Data bus is PORTB ---------------------------------
----------------*/


 #define F_CPU 8000000L    // 8 mHZ #include <avr/io.h>

#include <util/delay.h>

//-----------LCD Functions declaration ---------------
-   - - - - - - - - - - -
-   - - - - - - - - - - -
-    - - - - -- - - - - -

 //-------------------------------------------------

 //change here to match your

target board
```

```
---------- --------  ------------

--------  ------------ ----------

//-----------------------------------------------------//

/……MAIN FUNCTION ..... //

 --------      ------------    -----------------
--------      ------------    -----------------
--------      ------------    -----------------

 // ------------- Initialize the LCD driver ------------------- //

 --------      ------------    -----------------
 --------      ------------    -----------------
// ------------------------------------------------------------ //

 --------- Write a command instruction to the LCD ------------

 // ------------------------------------------------------------ //
 --------      ------------    -----------------
 --------      ------------    -----------------


 // ------------------------------------------------------------ //

 ------------- Write one byte of data to the LCD -----------

 // ------------------------------------------------------------

 --------      ------------    -----------------
 --------      ------------    -----------------

// ------------------------------------------------------ //

 --- Display a string at the specified row and column

 // ------------------------------------------------

 --------      ------------    -----------------
 --------      ------------    -----------------

// ------------------------------------------------------ //

 ---- Position the LCD cursor at "row", "column".-------

 //-------------------------------------------------

 --------      ------------    -----------------
 --------      ------------    -----------------
```

//----------

OUTPUT

RESULT

	Signature of Lab in Charge	Remarks
Readiness to do experiment		
Completion of Experiment		