R. V. College Of Engineering®

(Autonomous Institution Affiliated to VTU, Belagavi)

Department of Information Science & Engineering

Bengaluru, Karnataka– 560059

Lab Manual for IV Semester B.E

Microcontrollers and Embedded Systems

16IS45

Faculty In-charge

Prof. Raghavendra Prasad S.G
Assistant Professor

USN:

NAME:

Academic Year: **2018 – 19**

R. V. College Of Engineering®

(Autonomous Institution Affiliated to VTU, Belagavi)

Department of Information Science & Engineering

Bengaluru, Karnataka– 560059

C E R T I F I C A T E

This is to certify that Mr./Ms……………………………………………………………………………... USN …………………………………. of IV Semester Department of Information Science and Engineering Branch has satisfactorily completed the course of experiments in Microcontrollers and Embedded Systems laboratory prescribed by the …………………………… university in the academic year 2018 – 2019.

MARKS	
MAXIMUM	OBTAINED
RECORD (40Marks)	
TEST (10 Marks)	
Total (50 Marks)	

Signature of the Student **Signature of the Faculty In-Charge** **Signature of the HoD**

Vision, Mission, PEO, PO and PSO of the department

Vision

To be the hub for innovation in Information Science & Engineering through Teaching, Research, Development and Consultancy; thus make the department a well known resource center in advanced sustainable and inclusive technology.

Mission

ISE1: To enable students to become responsible professionals, strong in fundamentals of information science and engineering through experiential learning.

ISE2: To bring research and entrepreneurship into class rooms by continuous design of innovative solutions through research publications and dynamic development oriented curriculum.

ISE3: To facilitate continuous interaction with the outside world through student internship, faculty consultancy, workshops, faculty development programmes, industry collaboration and association with the professional societies.

ISE4: To create a new generation of entrepreneurial problem solvers for a sustainable future through green technology with an emphasis on ethical practices, inclusive societal concerns and environment.

ISE5: To promote team work through inter-disciplinary projects, co-curricular and social activities.

Program Educational Objectives (PEOs)

PEO1: To provide adaptive and agile skills in Information Science and Engineering needed for professional excellence / higher studies /Employment, in rapidly changing scenarios.

PEO2: To provide students a strong foundation in basic sciences and its applications to technology.

PEO3: To train students in core areas of Information science and Engineering, enabling them to analyze, design and create products and solutions for the real world problems, in the context of changing technical, financial, managerial and legal issues.

PEO4: To inculcate leadership, professional ethics, effective communication, team spirit, multi-disciplinary approach in students and an ability to relate Information Engineering issues to social and environmental context.

PEO5: To motivate students to develop passion for lifelong learning, innovation, career growth and professional achievement.

Program Outcomes (PO)

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **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.

3. **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.

4. **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.

5. **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.

6. **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.

7. **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.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **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.

11. **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.

12. **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 Outcome (PSO)

PSO-1 Recognize and appreciate the principles of theoretical foundations, data organization, data communication, security and data analytical methods in the evolving technology

PSO-2 Learn the applicability of various system softwares for the development of quality products in solving real-world problems with a focus on performance optimization

PSO-3

Demonstrate the ability of team work, professional ethics, communication and documentation skills in designing and implementation of software products using the SDLC principles

R. V. College Of Engineering®

(Autonomous Institution Affiliated to VTU, Belagavi)

Department of Information Science & Engineering

Bengaluru, Karnataka– 560059

Microcontrollers and Embedded Systems - 16IS45

PARTICULARS OF THE EXPERIMENT

Week	Program No.	Program	Page No.	Date of Execution	Marks (10)
1	1a	8051 ALP programs to perform block data transfer and searching operations	1		
2	2a	8051 ALP programs to perform Arithmetic (addition/subn/mult/divn) Operations	6		
	3a	8051 ALP programs to perform number conversions, binary to BCD,binary to ASCII	11		
3	4a	8051 ALP programs to compute average & maximum/minimum values	16		
	5a	8051 ALP program to perform sorting operations	21		
4	1b	8051 Embedded C programs to Interface Logical Controller	31		
	2b	8051 Embedded C programs to Interface Seven Segment Display	41		
5	3b	8051 Embedded C programs to interface Stepper Motor Module	46		
6	4b	8051 Embedded C programs to Interface DAC Module	51		
	5b	8051 Embedded C programs to Interface Keyboard Module	57		
7	6a	ARM Assembly Language Programs	66		
8	6b	8051 Embedded C programs to Interface LCD	70		
9		Mini Project - Hardware & Software	72		
				TOTAL	

Scheme of Continuous Internal Evaluation for Practical:

LAB INTERNALS	
RECORD:	/ 40 Marks
TEST:	/ 10 Marks
TOTAL:	/ 50 Marks

Scheme of Semester End Examination for Practical:

Total Marks:50

This includes:

- Program Write-up, Execution, and Results : 40
- Viva Voce :10

R. V. College Of Engineering®

(Autonomous Institution Affiliated to VTU, Belagavi)

Department of Information Science & Engineering

Bengaluru, Karnataka– 560059

General Guidelines

- Use Keil for editing ALP/Embedded C programs
- Use only Flash magic to burn the hex file to microcontroller.
- Students are encouraged to try the suggested modifications for all experiments
- Use the components and hardware with care.

Do's and Don'ts in the Laboratory

Do's……………..

- Come prepared to the lab with the necessary algorithms, which helps in getting better solutions.
- Use the computers for academic purposes only.
- Following the lab exercise cycles as per the instructions given by the faculty.
- Keep the chairs back to their position before you leave.
- Handle the computer and the kits with care.
- Keep your lab clean.

Don'ts……………..

- Coming late and leaving the lab early.
- Move around in the lab during the lab session.
- Download or install any software onto the computers.
- Tamper system files or try to access the server.
- Write Data sheets or Records in lab.
- Change the system assigned to you without the notice of lab staff.
- Carrying CDs, Pen Drives and other storage devices into lab.
- Using others login ids.

Mapping of CO with Lab programs

Course Outcomes

1 Acquire the knowledge of architecture of Microprocessors and Microcontrollers.

2 Develop skill in simple program writing for micro controllers assembly level language and Embedded C.

3 Apply acquired knowledge to design for interface and programming.

4 Analyze the design and implement for applications.

Program No.	**Program**	**CO1**	**CO2**	**CO3**	**CO4**
1a	8051 ALP programs to perform block data transfer and searching operations	**Y**	**Y**		
2a	8051 ALP programs to perform Arithmetic (addition/subn/mult/divn) Operations	**Y**	**Y**		
4a	8051 ALP programs to compute average & maximum/minimum values	**Y**	**Y**		
3a	8051 ALP programs to perform number conversions, binary to BCD,binary to ASCII	**Y**	**Y**		
5 a	8051 ALP program to perform sorting operations	**Y**	**Y**		
3 b	8051 ALP/Embedded C to interface Stepper Motor Module	**Y**	**Y**	**Y**	**Y**
1 b	8051 ALP/Embedded C program to Interface Logical Controller	**Y**	**Y**	**Y**	**Y**
2b	8051 ALP/Embedded C to Interface Seven Segment Display	**Y**	**Y**	**Y**	**Y**
5 b	8051 ALP/Embedded C to Interface Keyboard Module	**Y**	**Y**	**Y**	**Y**
4b	8051 ALP/Embedded C to Interface DAC Module	**Y**	**Y**	**Y**	**Y**
6 a	ARM assembly language programs	**Y**	**Y**		
6 b	Interface Graphics LCD and I2C device to ARM Microcontroller	**Y**	**Y**	**Y**	**Y**
Marks Distribution for each CO's					

Rubrics for Evaluation

Each program is evaluated for 10 marks.

Sl no	Criteria	Measuring methods	Excellent	Good	Poor	CO
Lab Record Write-up and Execution Rubrics (Max: 6 marks)						
1	**Understanding of problem and requirements (2 Marks)**	Observations	Student exhibits thorough understanding of program requirements and applies ALP for Embedded C for 8051 concepts. **(2M)**	Student has sufficient understanding of program requirements and applies ALP / Embedded C for 8051 concepts. **(1.5M - 1M)**	Student does not have clear understanding of program requirements and is unable to apply ALP for Embedded C for 8051 concepts. **(0M)**	CO1
2	**Design & Execution (2Marks)**	Observations	Student demonstrates the design & execution of the program with optimized code with all the modifications and test cases handled. **(2M)**	Student demonstrates the design & execution of the program without optimization of the code and handles only few modifications and few test cases. **(1.5M - 1M)**	Student has not executed the program. **(0M)**	CO3, CO4
3	**Results and Documentation (2Marks)**	Observations	Documentation with appropriate comments and output with observations is covered in manual. **(2M)**	Documentation with only few comments and only few output cases is covered in manual. **(1.5M - 1M)**	Documentation with no comments and no output cases covered in manual. **(0M)**	CO1
Viva Voce Rubrics (Max: 4 marks)						
1	**Conceptual Understanding (2 Marks)**	Viva Voce	Explains related architecture & Assembly language programming / Embedded C related concepts involved. **(2M)**	Adequately explains architecture & Assembly language programming / Embedded C related concepts involved. **(1.5M - 1M)**	Unable to explain the concepts. **(0M)**	CO1, CO2
2	**Use of appropriate Design Techniques (2 Mark)**	Viva Voce	Insightful explanation of appropriate design techniques for the given problem to derive solution. **(2M)**	Sufficiently explains the use of appropriate design techniques for the given problem to derive solution. **(1.5M - 0.5M)**	Unable to explain the design techniques for the given problem. **(0 M)**	CO4

The 8051 Microcontroller was designed in 1980's by Intel. Its foundation was on Harvard Architecture and was developed principally for bringing into play in Embedded Systems.

Microcontroller 8051 block diagram is shown below.

8051 MICROCONTROLLER

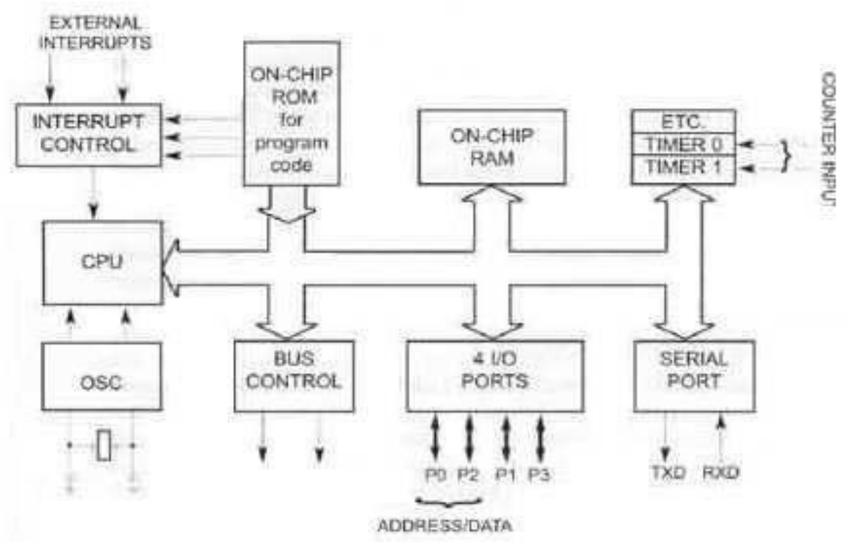


Fig : Inside the 8051 Microcontroller Block Diagram

Features

8051 is a 8 bit microcontroller from Intel, which has inbuilt

- Program / Code Memory – used to store instructions and constant data. 4K bytes of on- chip ROM
- Data Memory / RAM – 128 bytes, used to store variables, stack and to represent registers
- Programmable Input / Output bit addressable ports – 4 ports P0, P1, P2, P3, total 4 * 8 = 32 I/O pins.
- 2 Programmable 16 bit timers/counters, used for generating time related signals/ waveforms, for counting of events, without causing overhead to Microcontroller.
- Microcontroller can communicate to PC, using serial communication, full duplex – for uploading / downloading of data using the inbuilt serial port, so that lot of programming burden of microcontroller is reduced.

Addressing Modes

1. Immediate addressing mode ; Ex: MOV A,#05H
2. Register addressing mode ; Ex: MOV A,R0
3. Direct addressing mode : Ex: MOV A, 30H
4. Register Indirect addressing mode Ex: MOV A,@R0
5. Indexed Addressing mode: Ex: MOVC A,@A+DPTR

8051 Assembly Language Programs – Part A

Experiment No 1a:

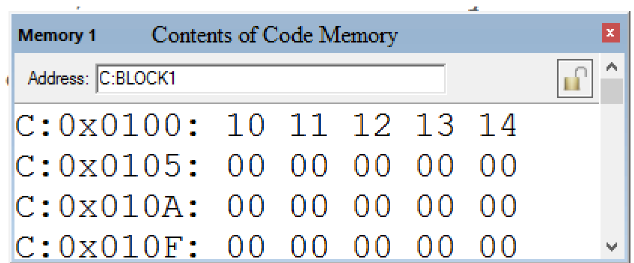
i. **Write an 8051 ALP to transfer block of data from code memory to data memory.**

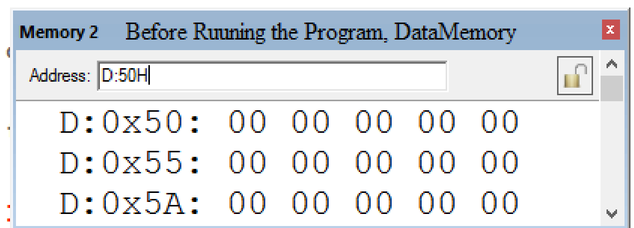
ii. **Write an 8051 ALP to perform linear search of n 8-bit numbers**

Program with the comments:

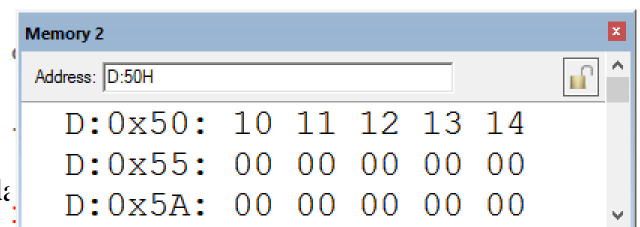
i. **Write an 8051 ALP to transfer block of data from code memory to data memory**

```
ORG 0000H
; Setup counter
   MOV R1, #05H          ;    Number of elements
; Setup Pointer
   MOV R0, #50H
;    Pointer to 50H memory location (Data Mem)
   MOV DPTR, #0100H
;    Pointer to 100 H memory location (Code Mem)
L1:   CLR A
; Do the data transfer
   MOVC A,@A+DPTR
   MOV @R0, A
; Update the pointer
   INC DPTR
   INC R0
; Check for completion of the count
   DJNZ R1, L1
; Store the data in the code memory
ORG 0100H
BLOCK1 : DB 10H,11H,12H,13H,14H
END
```





After running the Program, Data Memory contents



Expected Output:

The specified blocks of data should be moved from code to da

Output & Observations:

Write the memory location addresses with contents, before the program and after the program. Modify the program for different set of data and memory addresses, document the result.

Program with Comments:

ii. Write an 8051 ALP to perform linear search of n 8-bit numbers

```
        SRCH      EQU  11H
        RESULT  EQU  70H
        N           EQU    5

        ORG 0000H

        SJMP AHEAD                    ; data stored along with the code, hence jump over data

DATA1: DB   24H,45H,72H,30H,10H

AHEAD :
        MOV     DPTR, #DATA1       ; Set Up Dptr And Data
        MOV    R3,#N            ; Set Up Counter And Search Element

        ;Perform Operation

        CLR A
CONT:
        MOVC  A,@A+DPTR
        CJNE    A,#SRCH,NOTFND
        MOV     RESULT,#0FFH
        SJMP    DONE
NOTFND:                    ; Update Pointer And Check For Counter
        INC       DPTR
        DJNZ    R3,CONT
        MOV     RESULT,#0FH
DONE:
        SJMP $

        END
```

Expected Output:

The given numbers should be sorted in the ascending order.

Output & Observations:

Write the memory location addresses with contents, before the program and after the program. Modify the program for different set of data and memory addresses, document the result.

Assignment Programs:

1. **Find the biggest/smallest of two numbers**
   (store the numbers in code memory, and store the result in data memory)

2. **Find the biggest/smallest of three numbers**
   (store the numbers in code memory, and store the result in data memory)

3. **Modify the program number 2, to find the biggest of three numbers which are stored in RAM locations 45H, 46H and 47H. Load the biggest number in R2.**

<table>
<tr><th rowspan="3">Program No.</th><th colspan="3">Marks for Execution
(7)</th><th colspan="2">Marks for Viva voce
(3)</th><th rowspan="3">TOTAL
(10)</th><th rowspan="3">Signature of the Faculty</th></tr>
<tr><th colspan="3">Rubrics</th><th colspan="2">Rubrics</th></tr>
<tr><th>Understanding of problem
(2)</th><th>Execution
(3)</th><th>Results and Documentation
(2)</th><th>Conceptual Understanding and Communication of Concepts
(2)</th><th>Use of appropriate Design Techniques
(1)</th></tr>
<tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

Program No. 1a

Paste your DATA SHEET here

[Page intentionally left blank, student to paste his/her data sheet after evaluated by the staff in-charge]

Experiment No 2a:

AIM: Write an 8051 ALP program to perform Arithmetic operations (addition / subtraction / multiplication / division operations). ALP to implement simple calculator is given below

```
OPTION      EQU   00H  ; 0-Addition ,1-Subtraction, 2-Multiplication, 3-Division
NUM1        EQU   02H  ; First number
NUM2        EQU   07H  ; Second number

        ORG 0000H
        MOV   R0, #OPTION
        CJNE  R0, #00,CKSUB
        MOV   A, #NUM1
        MOV   B, #NUM2
        ADD   A, B                    ; Perform Addition
        MOV   B, #00                  ; B Has Carry
        JNC   SKIP
        MOV   B, #01H
SKIP:
        SJMP  LAST
CKSUB:
        CJNE  R0,#01,CKMUL
        MOV   A, #NUM1
        MOV   B, #NUM2
        CLR C                         ; Reset Borrow Flag
        SUBB  A , B                   ; Perform Subtraction
        MOV   B, #00                  ; B Indicates Borrow
        JNC   SKIP1
        MOV   B, #0FFH                ; FF Indicates Negative Number
SKIP1:
        SJMP  LAST
CKMUL:
        CJNE  R0, #02, CKDIV
        MOV   A, #NUM1
        MOV   B, #NUM2                ; Perform Multiplication
        MUL   AB                      ; 16 bit product in AB with A having lower byte
        SJMP  LAST
CKDIV:
        CJNE  R0, #03, OTHER
        MOV   A, #NUM1
        MOV   B, #NUM2                ; Perform Division
        DIV   AB                      ; Quotient in A & remainder in B
        SJMP  LAST
OTHER:
        MOV   A, #00                  ; Store 00 for invalid option
        MOV   B, #00
LAST:
        MOV   R0, #70H                ; Answer is stored in the Data memory 70h  and 71h
        MOV   @R0, A
        INC   R0
        MOV   @R0, B
HERE:
        SJMP  HERE
        END
```

Expected Output:

The result of addition/subtraction/multiplication/division are observed at the memory locations 70h and 71h, by changing the option 0 to 3.

Output & Observations:

Write the memory location addresses with contents, before the program and after the program. Modify the program for two different sets of numbers for each of the option.

Assignment Programs:

1. Add the first 20 natural numbers and store the sum in a RAM location.

2. Add four 16-bit numbers which are in consecutive memory locations, assuming the sum does not go above 16 bits.

3. The selling price of 5 items are stored in ROM locations 0100H onwards. The corresponding cost prices are entered in RAM locations from 40H onwards. Calculated the average profit of the five items.

Program No.	Marks for Execution (7)			Marks for Viva voce (3)		TOTAL (10)	Signature of the Faculty
	Rubrics			Rubrics			
	Understanding of problem (2)	Execution (3)	Results and Documentation (2)	Conceptual Understanding and Communication of Concepts (2)	Use of appropriate Design Techniques (1)		
1							
2							

Program No. 2a

Paste your DATA SHEET here

[Page intentionally left blank, student to paste his/her data sheet after evaluated by the staff in-charge]

Experiment No 3a:

AIM: Write an 8051 A LP to perform following number conversions

- **i. binary to BCD**
- **ii. binary to ASCII.**

Problem Description:

8051 ALP program to perform Binary to BCD number conversion.
Binary number to be entered in location 70H. Store the BCD from location 75H

Algorithm

1. Move the hex data to be converted to A-accumulator.
2. Move 10 to B register and divide A, store the remainder (in B, BCD number) in unit's place in memory
3. Decrement memory location
4. Move 10 to B register and divide A (which contains quotient from step2), store the remainder (in B- BCD number) in tens place in memory.
5. A (quotient) , contains BCD digit (hundreds place)

.**Program:**

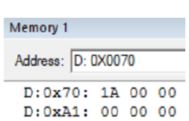
```
        ORG   0000H
        MOV   70H, #0FFH   ;
        MOV   R0, #75H
        MOV   A, 70H       ; Get hex number / binary number
        MOV   B,#10        ; 10 is the base of the destination number system (BCD)
        DIV   AB           ; divide by 10 (0AH) to extract the decimal digits
        XCH   A,B
        MOV   @R0,A        ; Store the remainder (in B)  in units place
        XCH   A, B
        MOV   B, #10       ; divide by 10(0Ah) to extract next BCD digit
        DIV   AB
        DEC   R0
        XCH   A,B
        MOV   @R0, A       ; Store the remainder (in B) in tens place
        XCH   A,B
        DEC   R0
        MOV   @R0,A        ; Store the quotient (in A) in hundreds place
HERE:   SJMP HERE

        END
```

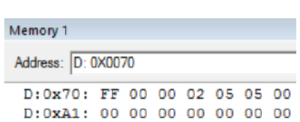
Expected Output:

I/P – FF O/P - 255 (stored in the memory from 75H – 05,05,02)

Before Conversion



After Conversion



Output & Observations:

Write the memory location addresses with contents, before the program and after the program.
Modify the program for two sets of data.

Program :

Problem Description:

Write an 8051 ALP program to perform Binary to ASCII number conversion.
Binary number to be entered in location 70H. Store the BCD from location 75H

Algorithm :

1. Move the hexadecimal data to be converted to accumulator.
2. Get the lower nibble
3. If digit greater than 09,(for A-F)
      add 07h & 30h
   Else (i.e., for 0-9)
      add only 30h
5. Store the converted ASCII value, of lower nibble
6. Get the higher nibble
7. If digit greater than 09,(for A-F)
      add 07h & 30h
   Else (i.e., for 0-9)
      add only 30h
9. Store the converted ASCII value, of upper nibble

Program:

```
        ORG 0000H
        MOV   A, 70H //2-digit number to be converted is given in data memory 70h
        ANL    A,  #0F0H //obtain upper digit
        SWAP  A //bring to the units place
        CJNE    A,#0AH,CONTINU1
CONTINU1:
        JNC    NEXT
        ADD  A, #30H
        JMP    STORE1
NEXT :
        ADD   A,#37H
STORE1:
        MOV   71H, A
        MOV   A, 70H
        ANL    A,  #0FH //obtain LOWER digit
        CJNE   A, #0AH,CONTINU2
CONTINU2:
        JNC    LAST
        ADD  A,#30H
        JMP    STORE2
LAST:
        ADD   A, #37H
STORE2:
        MOV   72H, A
HERE: SJMP HERE
        END
```

Expected Output:

Input – 10 ; Output – 31h, 30h
Input - 0A; Output – 30h, 41h

Output & Observations:

Write the memory location addresses with contents, before the program and after the program. Modify the program for two sets of data.

Assignment Program:

1. 50 bytes are stored from locations 34H onwards. Find out how many of these bytes are zero.

2. Assume that 5 BCD data items are stored in RAM locations starting at 40H. Write a program to find the sum of all the numbers. The result must be in BCD.

3. Implement the 8051 ALP using look up table, to convert single digit(unpacked) decimal number to corresponding ASCII number.

<table>
<tr><th rowspan="3">Program No.</th><th colspan="3">Marks for Execution
(7)</th><th colspan="2">Marks for Viva voce
(3)</th><th rowspan="3">TOTAL
(10)</th><th rowspan="3">Signature of the Faculty</th></tr>
<tr><th colspan="3">Rubrics</th><th colspan="2">Rubrics</th></tr>
<tr><th>Understanding of problem
(2)</th><th>Execution
(3)</th><th>Results and Documentation
(2)</th><th>Conceptual Understanding and Communication of Concepts
(2)</th><th>Use of appropriate Design Techniques
(1)</th></tr>
<tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

Program No.3a

Paste your DATA SHEET here

[Page intentionally left blank, student to paste his/her data sheet after evaluated by the staff in-charge]

Experiment 4a:

AIM: Write an 8051 ALP to find the maximum and minimum values in the list of ‘n’ elements present in the code memory.

Algorithm:

Step 1: Initialization of variables
DPTR ← Address of memory, where elements are stored
MIN/MAX ← (DPTR) , assume the first element as MIN/MAX
(MIN/MAX are Data memory locations, can be used as variables)
R0 ← Number of elements – 1 ; initialising the counter for no. of operations

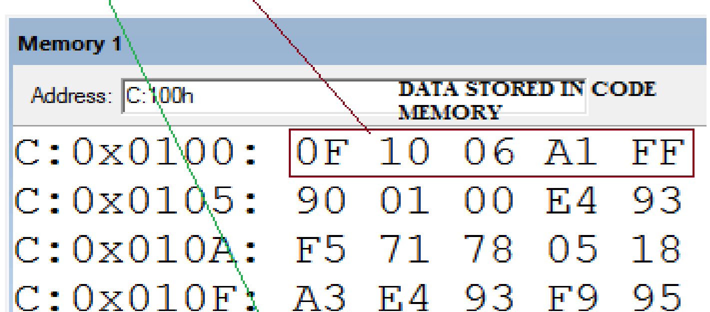
Step 2: Get the next number from memory
DPTR = DPTR + 1
A ← (DPTR)
R1 ← A , store temporarily, as SUBB instruction disturbs the value in A

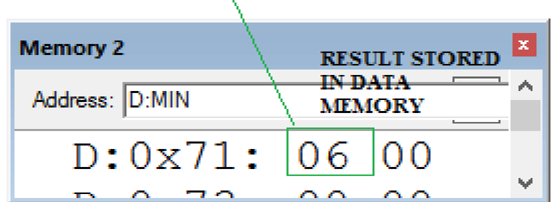
Step 3: Compare with the MIN/MAX and Update the variable
If (A – MIN/MAX) < 0 // >0 for MAX
MIN/MAX ← R1

Step 4: Check for the completion of all numbers
R0←R0-1,
If R0 ≠0 continue to step 2

Step 5: END

```
COUNT      EQU     05H
MIN        EQU     71H
           ORG      0000H
           JMP   AHEAD
           ORG      0100H
ELEMENTS  :   DB   0FH,10H,06H,0A1H,0FFH
AHEAD  : MOV    DPTR,#ELEMENTS
           CLR    A
           MOVC   A,@A+DPTR
           MOV    MIN,A
           MOV    R0,#COUNT
           DEC    R0
BACK:      INC    DPTR
           CLR    A
           MOVC   A,@A+DPTR
           MOV    R1,A
           SUBB   A,MIN
           JNC    SKIP
           MOV    MIN,R1
SKIP:      DJNZ   R0,BACK
           SJMP   $
           END
```





Modify the code to include MAX computation

Output & Observations:

Write the memory location addresses with contents, before the program and after the program.
Modify the program for two sets of data.

4a. Write an 8051 ALP program to compute average of n 8bit numbers

Assignment Programs:

1. Write a program to find the number of 1s in a given byte.

2. Assume RAM memory locations 40H-44H contain the daily temperature for five days. Check if any of the values is equal to 65, give its location to R4, otherwise make R4=0.

Program No.	Marks for Execution (7)			Marks for Viva voce (3)		TOTAL (10)	Signature of the Faculty
	Rubrics			Rubrics			
	Understanding of problem (2)	Execution (3)	Results and Documentation (2)	Conceptual Understanding and Communication of Concepts (2)	Use of appropriate Design Techniques (1)		
1							
2							

Program No.4a

Paste your DATA SHEET here

[Page intentionally left blank, student to paste his/her data sheet after evaluated by the staff in-charge]

Experiment 5a:

AIM: Write an 8051 ALP to implement Bubble sort.

Algorithm:

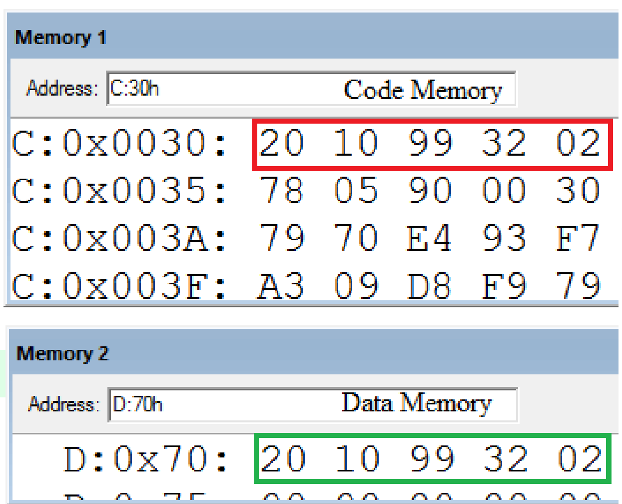
1. Store the elements of the array from the address 0030H (Code memory)
2. Move the array elements from code memory to data memory (Starting address 70H)
3. Initialize a pass counter (R1 ←#COUNT -1) with array size-1 count (for number of passes).
4. Load compare counter(No. Of passes) with pass counter contents
   R1←R2
   R0 ← Start address of the array, 70H
5. Store the current and the next array elements pointed by R0 in registers A and B respectively.
6. Subtract the next element from the current element.
7. If the carry flag is set (for ascending order) then exchange the 2 numbers in the array.
8. Decrement the compare counter and repeat through step 5 until the counter becomes 0.
9. Decrement the pass counter and repeat through step 4 until the counter becomes 0.

ALP Program

; first transfer the data from code memory to data memory

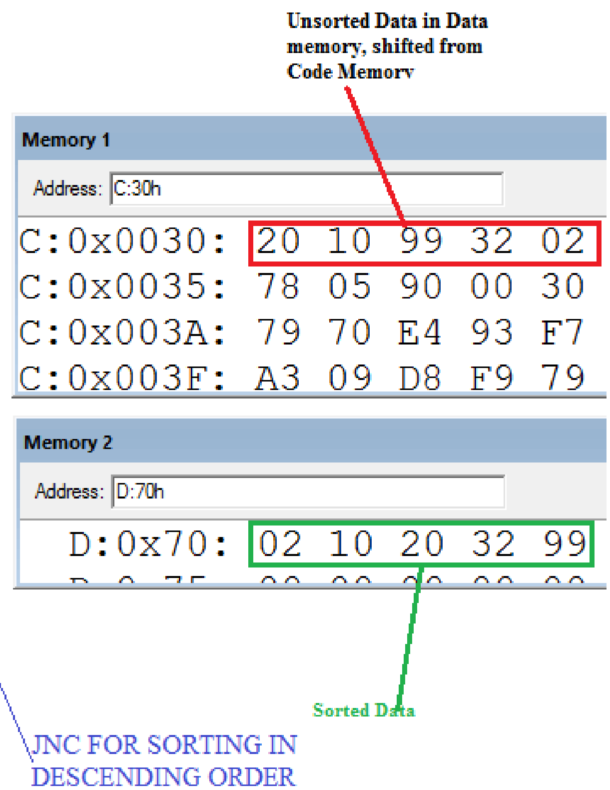
```
COUNT EQU 5
ORG 0000H
      SJMP START
ORG 0030H
      NUMS : DB 20H,10H,99H,32H,02H
START:
      MOV R0,#05H
      MOV DPTR,#NUMS
      MOV R1,#70H
NEXT:
      CLR A
      MOVC A,@A+DPTR
      MOV @R1,A
      INC DPTR
      INC R1
      DJNZ R0,NEXT
```

First Move Data from Code Memory to Data Memory starting at 70h



; Now sort the moved Data in data memory

```
      MOV R1,#COUNT
      DEC R1
AGAIN:
      MOV B,R1
      MOV R2,B
      MOV R0,#70H
UP:  MOV A,@R0
      INC R0
      MOV B,@R0
      CLR C
      SUBB A,B
      JC SKIP
      DEC R0
      MOV A,B
      XCH A,@R0
      INC R0
      MOV @R0,A
SKIP:
      DJNZ R2,UP
      DJNZ R1,AGAIN
HERE: SJMP HERE
      END
```



Output & Observations:

Write the memory location addresses with contents, before the program and after the program.
Modify the program for two sets of data and document the result

Assignment Program:

1. Write an ALP to compute GCD and LCM of two 8 bit numbers using procedures.
2. Write an ALP to compute Factorial of a number using procedures and using stack (assume result is limited to maximum of 8bits.

Program No.	Marks for Execution (7)			Marks for Viva voce (3)		TOTAL (10)	Signature of the Faculty
	Rubrics			Rubrics			
	Understanding of problem (2)	Execution (3)	Results and Documentation (2)	Conceptual Understanding and Communication of Concepts (2)	Use of appropriate Design Techniques (1)		
1							
2							

Program No.5a

Paste your DATA SHEET here

[Page intentionally left blank, student to paste his/her data sheet after evaluated by the staff in-charge]

8051 Embedded C -Hardware Programs [Part B]

Introduction

RVCE ALL-IN-ONE INTERFACE Card

R&D Labs of CSE Department, RVCE has taken up the Design & Development of “All In One – Multipurpose Interfacing Card” for PC for X86 Programming and product developments, **successfully manufactured and adopted at our department, also at many other colleges and** number of students successfully carried out their projects using this card.

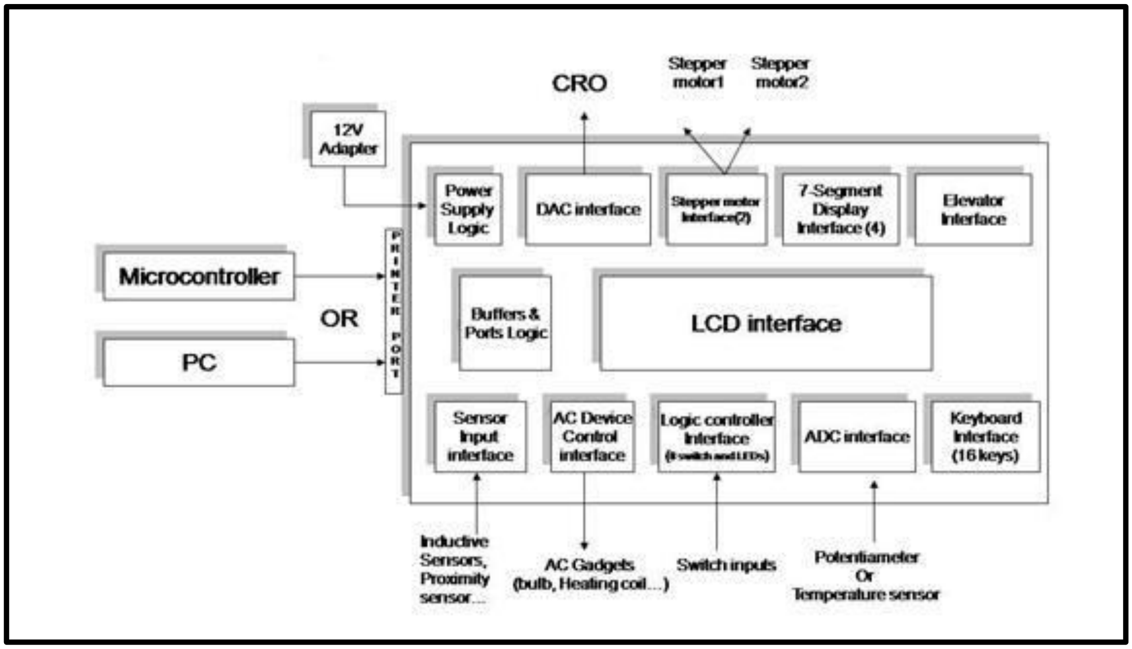


Figure : Block Diagram of RVCE ALL-IN-ONE Interface Card

Using RV All-In-One Interface Card,

- You can convert your PC to PC based product!! for number of applications
- To create interest and bring innovation among students by using the interface card to conduct different experiments and projects using their computer system

Features

- To Perform 80x86(usingPC) Interfacing Experiments, all in one board(Over 11 Module/logics Integrated)
- Interfaced through PrinterPort, so no 8255 Add-on Card required, so portability in conducting experiments on any PC
- To Build Prototype Products Using Software& HardwareFeatures
- To Learn H/W Programming-Using Assembly,C,C# and Java

Student can learn

80x86- Interfacing using Assembly and Turbo C.

80x51- Interfacing experiments using Assembly and Embedded C

The Different Interfacing Modules Integrated in RV ALL-IN-ONE Interface Board are….

- Logic Controller
- Seven Segment Display Module
- LCD Interface
- Stepper Motor Interface
- DAC Interface
- ADC Interface
- DC Motor Interface
- DC Solenoid
- Temperature Sensor Interface
- Ac Gadget Interface
- Industrial Sensors Input Interface
- Elevator Interface
- Keyboard Interface

Specifications

Power Input : 12V DC

Logic Input: 25 Pin D-type Female connector to interface to different logics, compatible to 25pin D-type printer port provided in the computers.(TTL compatible input/outputs- 5V logic 1,0V-logic 0)

8051 MICROCONTROLLER

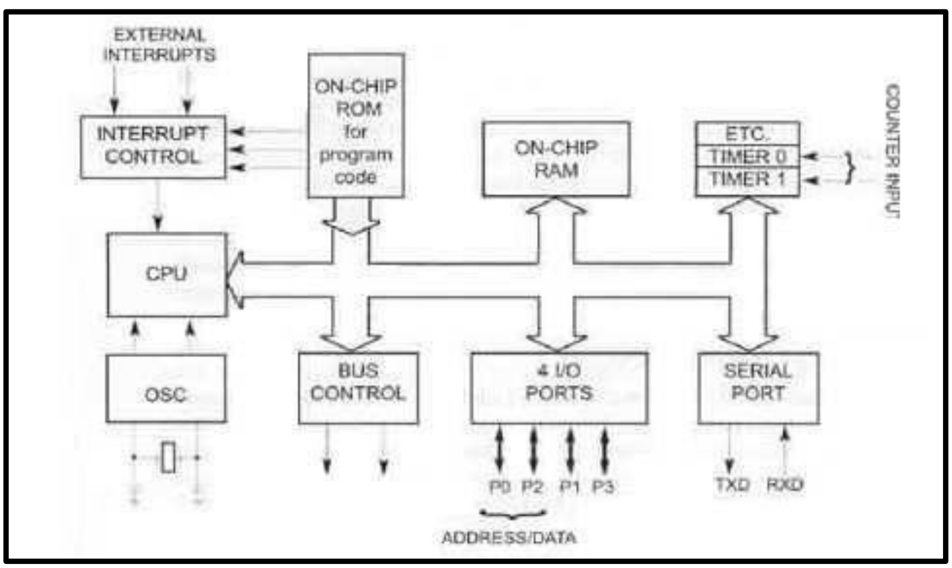


Figure : Inside the 8051 Microcontroller Block Diagram

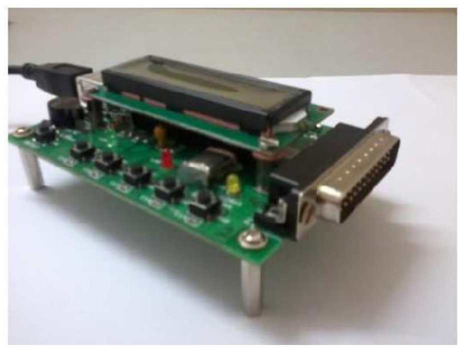
8051 is a 8 bit microcontroller from Intel, which has inbuilt

- Program / Code Memory – used to store instructions and constant data. 4K bytes of on-chip ROM
- Data Memory / RAM – 128 bytes, used to store variables, stack and to represent registers
- Programmable Input / Output bit addressable ports – 4 ports P0, P1, P2, P3, total 4 * 8 = 32 I/O pins.
- 2 Programmable 16 bit timers/counters, used for generating time related signals/ waveforms, for counting of events, without causing overhead to Microcontroller.

- Microcontroller can communicate to PC, using serial communication, full duplex – for uploading / downloading of data using the inbuilt serial port, so that lot of programming burden of microcontroller is reduced.

RV-USB Based 8051 Kit

This is USB based 8051 Compatible Microcontroller based Development Kit , designed & developed at R&D Labs, CSE Dept., to enable students to conduct Microcontroller based interfacing experiments and build prototype projects using RV-All-In-One Interface card.



This development kit has the following features…

- Compatible with RV-All-In-One Interface card
- Compatible with USB interface of PC(driver provided)
- Compatible with Keil Microvision3/ FlashMagic (auto loading facility with SW Reset provided)
- LCD ,Buzzer ,RTC ,LED interfaces provided
- Software reset, 5 function keys
- 25 pin D Type, Centronics Compatible (used in PC’s) printer port
- Effectively used for Microcontroller based Program learning, Interface learning and product development

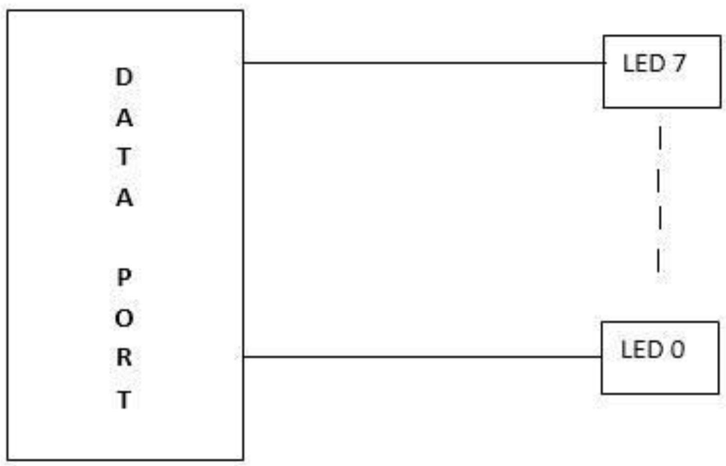
Logic Controller Interface

Description:

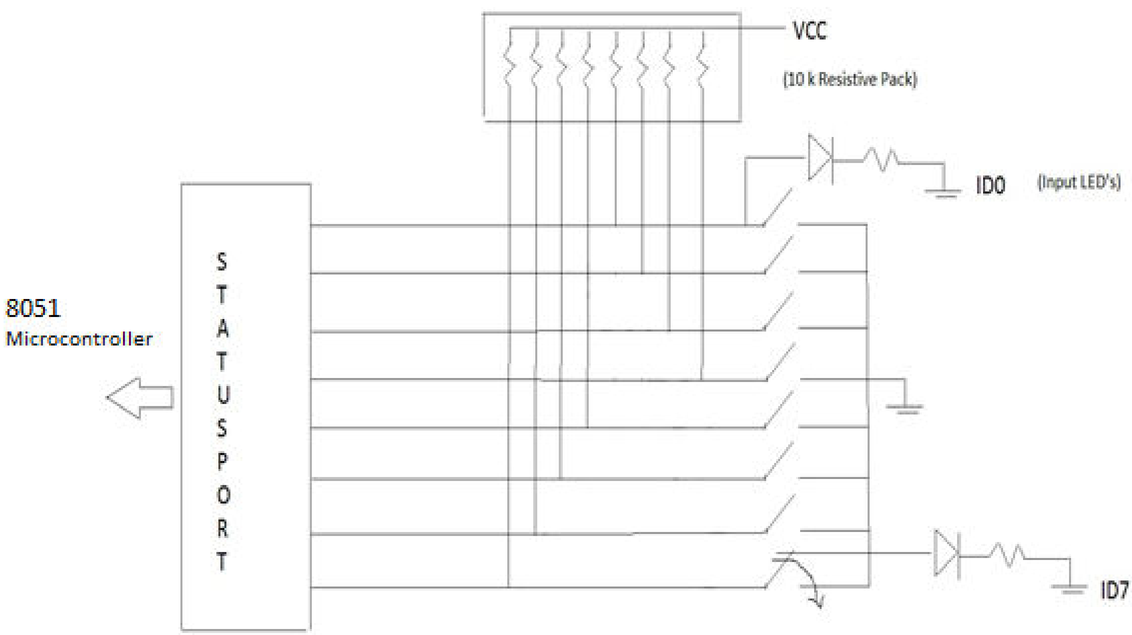
It comprises of

- 8 leds, connected to output port, through current limiting resistors
- 8 switches connected to input port, through pull up resistors

Sending 1 on data port pins, sets the corresponding led ON, sending 0 clears the corresponding led.

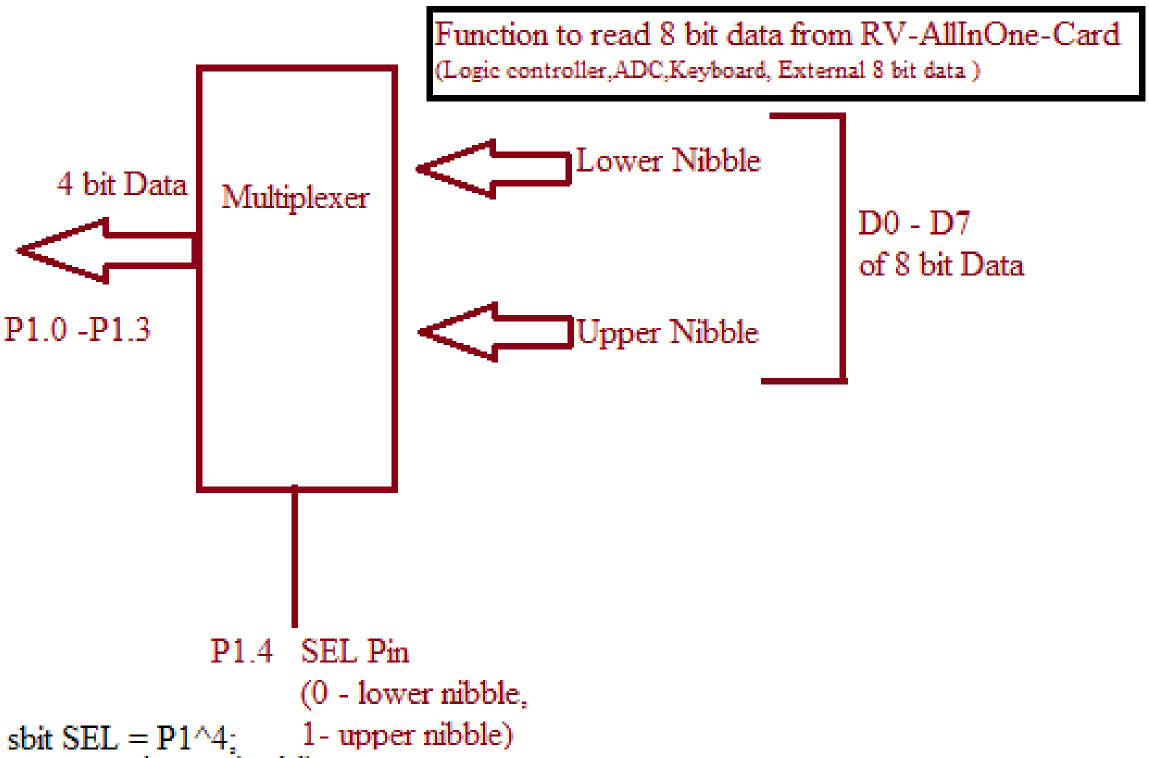


Note : P0 is used as DataPort, when 8051 kit is connected to RV-All-In-One-Card



When switch is kept in ON position, we receive 1, else 0 is read to microcontroller through input port.

Since RV AllInOne board provides one 4 bit input port (designed to provide compatibility with PC printer ports), to read 8 bit data we require to read two times two nibbles, this is made possible using multiplexer, whose input is 8 bits, (connected to different logics like logic controller switches, ADC and keyboard) and output is 4 bits. SEL line is used to select which nibble of the multiplexer to read, SEL=0 means lower nibble, SEL=1 means, upper nibble.




```
sbit SEL = P1^4;
tByte readInput(void)
{
        tByte temp = 0;
        SEL = 0;                              // read the lower nibble
        temp = P1 & 0x0F;
        SEL = 1;                              // read the upper nibble
        temp = (P1 & 0x0F) << 4 | temp;       // shift 4bits, for upper nibble
        return temp;                          //  and then OR with lower nibble
}
```

This routine would not have been required, if all 8 bits are connected to port, then

temp = P0 ;

would be sufficient.

Experiment No – 1b(i)

Aim: Write an Embedded C program to implement Decimal UP / Decimal Down / Ring Counter using Logic Controller Interface module.

Embedded C Program:

```
#include <reg51.h>
typedef  unsigned  char  tByte;        //8 bits
typedef  unsigned  int    tWord;       //16 bits
void delayMs(tWord);                //Delay function
sbit key1 = P3^2;
sbit key2 = P3^3;
sbit key3 = P3^4;
int main(void)
{
        tByte count=0,i,temp;
        tWord delay = 1000;           //Delay in milliseconds used for counting
        P0 = 0x00;                //Clear all LEDs
        while(1)
        {
          /*Program while loop, code in this loop will run continuously. When one of the
          keys is pressed, run the appropriate counter and exit back to this loop to check for
          key press again. When a key is pressed, the input value to the microcontroller will
          be 0.*/

          if(key1 == 0)
          {
                /* Decimal Up counter loop, use a for loop to increment the counting variable
                from 0 to 99, and output the number to P0, the output port used for the interface
                module.To display as decimal in the leds, first take the first digit of  the number,
                next take the second digit and set it as the upper nibble  by first left shifting it 4
                times and then ORing it. */

             for(count = 0 ; count <= 99 ; count++)
             {
                P0 = (count / 10) << 4 | count % 10;
                delayMs(delay);
             }
             P0=0x00;    //Turn all LEDs off
          }
          if(key2 == 0)
          {   //Decimal Down counter loop, works similar to the up counting loop, but
             //the for loop is used to decrement from 99 to 0
             for(count = 99 ; count >= 0 ; count--)
             {
                P0 = (count / 10) << 4 | count % 10;
                delayMs(delay);
             }

             P0=0x00;    //Turn all LEDs off
          }
```

```
            if(key3 == 0)
            {
                while(1)
                {
                    for(i = 0; i < 8; i++)
                    {
                        P0 = 0x01<<i;     // 0x01 means 0000 0001 bit pattern
                        delayMs(delay);
                    }
                }
            }
          }
}
void delayMs(tWord x)
{

          tWord i;
          while(x--)
            for(i=0;i<120;i++);
}
```

Modification:

- Change the maximum/minimum value used in the delay counter
- Change the ring counter logic to glow and shift two LEDs instead of one
- Change the direction in the ring counter

Output & Observations:

Experiment No – 1b(ii)

Aim: Write an Embedded C program to read the status of 8 inputs bits from 8bit switch and display FF' if it is even parity otherwise display 00. Also display number of 1's in the input data on the LED outputs, using Logic Controller interface module.

Embedded C Program:

```
#include<reg51.h>

typedef unsigned char tByte;
typedef unsigned int  tWord;

sbit SEL = P1^4;

void delayMs(tWord);        //Delay function
tByte readInput(void);      //Read 8 bits from input port
tByte countOnes(tByte);     //Returns number of 1s in the argument

int main(void)
{
        tByte temp,count;
        while(1)
        {
          temp = readInput();       // read the 8 bit data from logic controller
          count = countOnes(temp); // count the number of 1's in the 8 bit data

          if(count % 2 == 0)        // logic to check EVEN or ODD parity
                P0 = 0xFF;           // display all 1's for EVEN parity
          else
              P0 = 0x00;           // display all 0's for ODD parity

          delayMs(1000);
          P0 = count;              // now display count of 1's for next 1 second
          delayMs(1000);
        }
}

tByte countOnes(tByte x)
{
        tByte i,count = 0;
        for(i = 0 ;i < 8 ;i++)             // loop to check 8 different bits of a number
        {
          if(x & (0x01<<i))
              count++;         // keep incrementing whenever any bit is found as 1
        }
        return count;              //return count of 1's in a given number
}
```

```
tByte readInput(void)
{

        tByte temp = 0;
        SEL = 0;            // read the lower nibble
        temp = P1 & 0x0F;
        SEL = 1;            // read the upper nibble
        temp = (P1 & 0x0F) << 4 | temp;  // shift 4bits, for upper nibble
        return temp;                     //  and then OR with lower nibble

 }

void delayMs(tWord x)
{
        tByte i;
        while(x--)
          for(i=0;i<120;i++);
}
```

Modification:

- Display number of zeros with suitable delay

Output & Observations:

Experiment No – 1b(iii)

Aim: Write an Embedded C program to read the status of two 8-bit inputs (X and Y) and display the result X*Y using the interface module.

Embedded C Program:

```
#include<reg51.h>

typedef unsigned char tByte;      //8 bits
typedef unsigned int tWord;       //16 bits

void delayMs(tWord);              //Delay function
tByte readInput(void);            //Read 8 bits from input port

sbit SEL = P1^4;
sbit key1 = P3^2;
sbit key2 = P3^3;
sbit key3 = P3^4;

int main(void)
{
        tWord a = 0,b = 0,c = 0;  //equation is c = a * b
        P0 = 0x00;          //Clear all LEDs
        while(1)
        {
          if(!key1)
          {
          /* When key1 is pressed, the variable a is updated with the current value of the
                    input port. The updated value of a is displayed for half a second. */

              a = readInput();
              P0 = a;
              delayMs(500);
              P0 = 0x00;
              delayMs(500);
          }
          if(!key2)
          {
          /* When key1 is pressed, the variable b is updated with the current value of the
                    input port. The updated value of b is displayed for half a second. */

              b = readInput();
              P0 = b;
              delayMs(500);
              P0 = 0x00;
              delayMs(500);
          }
```

```
            if(!key3)
            {
            /* When key3 is pressed, the result of multiplication of the current values of a and
                        b is assigned to c. First the LSB of the result is displayed by ANDing the result
                        with 0xFF .Next the MSB is displayed by right shifting the result 8 times and
                        writing it to P0. */
                c = a*b;
                P0 = c & 0xFF;
                delayMs(1000);
                P0 = c >> 8;
                delayMs(1000);
                P0 = 0x00;
            }
          }
}

tByte readInput(void)
{

          tByte temp = 0;
          SEL = 0; // read the lower nibble
          temp = P1 & 0x0F;
          SEL = 1; // read the upper nibble
          temp = (P1 & 0x0F) << 4 | temp;   // shift 4bits, for upper nibble and then OR with lower nibble
          return temp;

}

void delayMs(tWord x)
{
          tByte i;
          while(x--)
            for(i=0;i<120;i++);
}
```

Modification:

- Also perform division for the two 8 bit inputs. Display the Quotient and Remainder with suitable delay.

Output & Observations:

Program No.	Marks for Execution (7)			Marks for Viva voce (3)		TOTAL (10)	Signature of the Faculty
	Rubrics			Rubrics			
	Understanding of problem (2)	Execution (3)	Results and Documentation (2)	Conceptual Understanding and Communication of Concepts (2)	Use of appropriate Design Techniques (1)		
1b (i)							
1b (ii)							
1b (iii)							

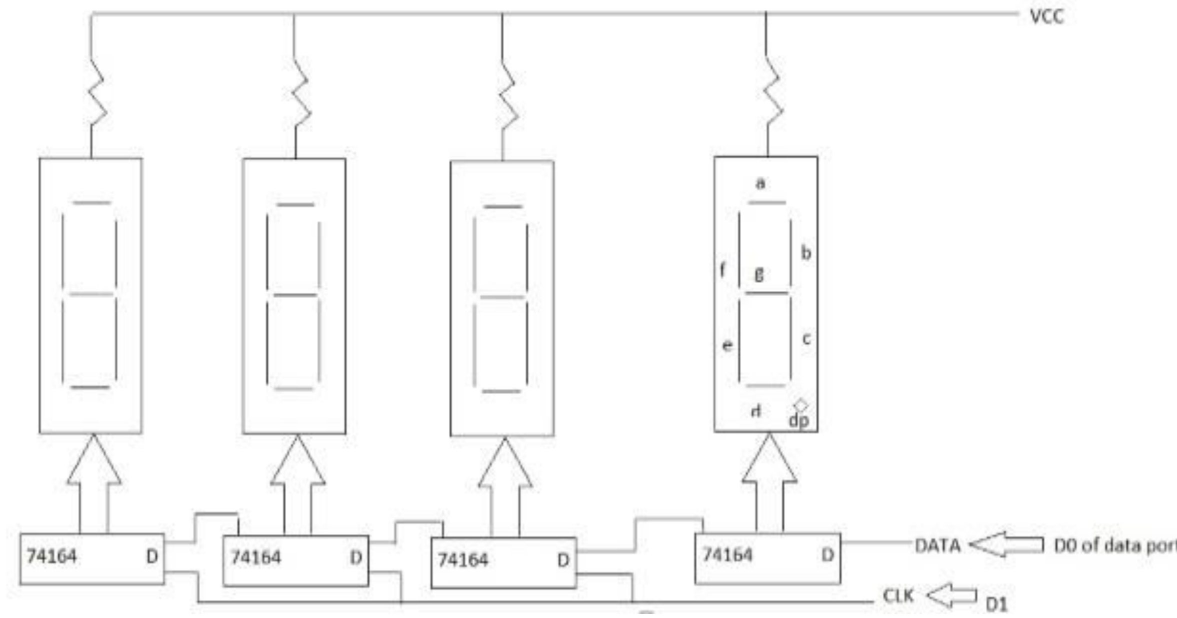
Program No.1b

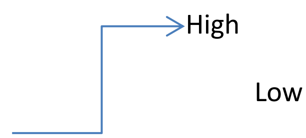
Paste your DATA SHEET here

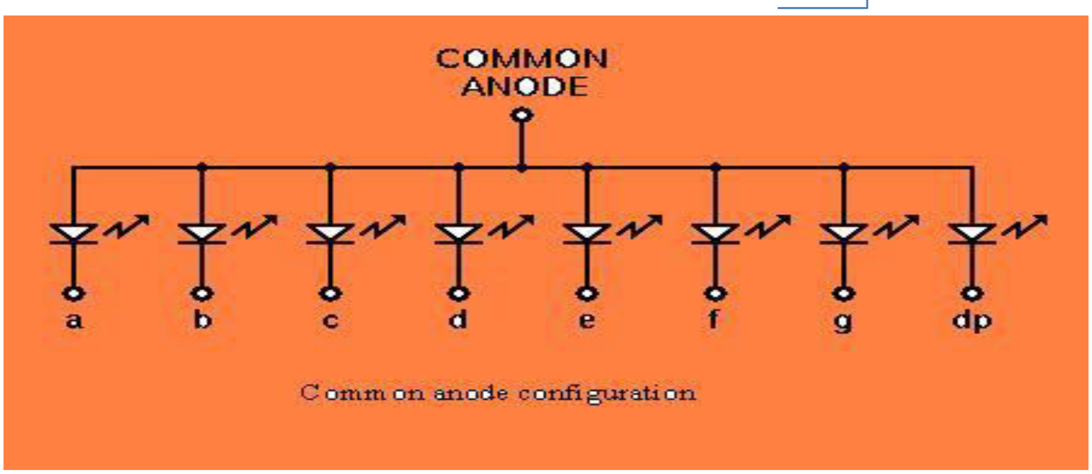
[Page intentionally left blank, student to paste his/her data sheet after evaluated by the staff in-charge]

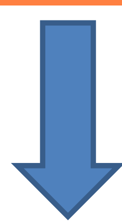
SEVEN SEGMENT DISPLAY MODULE

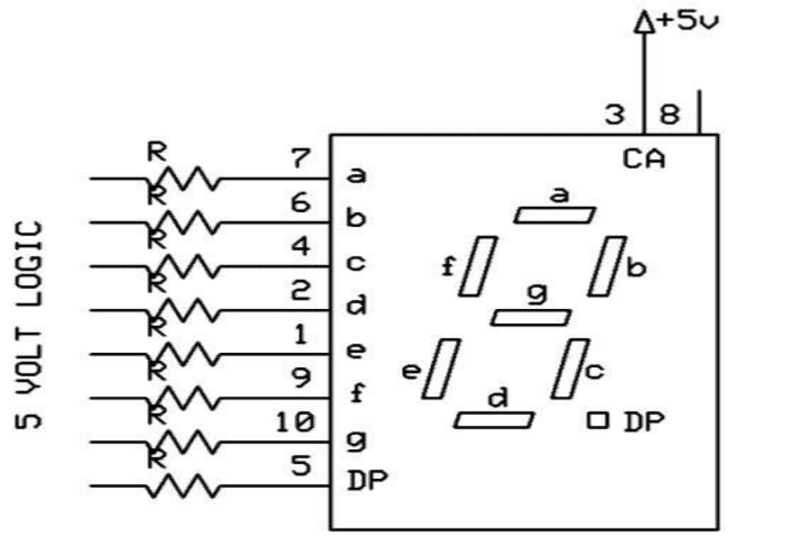
Serial In Parallel Out mode of Shift Register (74164) is used to send 8 bits of data to seven segment display. Seven segment display used is of common anode type i.e. we have to send 0 to make corresponding segment ON and 1 to make it OFF.









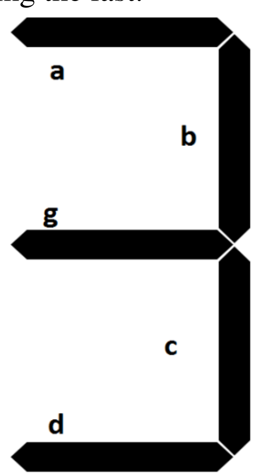


COMMON ANODE 7SEGMENT DISPLAY

To display 3, we have to send following bit pattern,

DP	G	f	e	d	c	b	a
1	0	1	1	0	0	0	0

This is B0 in hexadecimal. To send B0H we have to start sending the bits from MSB onwards i.e D7 first, D6 next and so on with D0 being the last.



Clock pulses are required to clock in the data, 8 clock pulses for one byte of data. As shift registers are cascaded, 8*4=32 clocks are required to clock in 4 bytes of data. To send “1234”, first we have to send ‘1’, then ‘2’,‘3’ and lastly ‘4’. All the shift registers are cascaded, the data is fed to the shift register using serial in parallel out method. The Data and Clock pins are connected to D0 and D1 of the output port respectively.

Experiment No – 2b

AIM: Write an Embedded C program to display messages “FIRE” & “HELP” on 4 digit seven segment display alternately with a suitable delay**.**

Embedded C Program:

```
#include <reg52.h>

typedef unsigned char tByte;  //8 bits
typedef unsigned int tWord;   //16 bits

sbit DAT = P0^0;
sbit CLK = P0^1;

void delayMs(tWord);           //Delay function
void writeSeg(tByte);          //Write the 8bit seven segment code to 7segment display
int main(void)
{
        int i = 0;
        tByte help[4] = {0x89,0x86,0xC7,0x8C};
        tByte fire[4]  = {0x8E,0xCF,0xAF,0x86};

        P0 = 0x00;
        while(1)
        {
          for(i=0;i<4;i++)
              writeSeg(help[i]);
          delayMs(1000);
          for(i=0;i<4;i++)
              writeSeg(fire[i]);
          delayMs(1000);
        }
}
void writeSeg(tByte x)
{
        tByte i;
        for(i = 0; i < 8; i++)
        {
          if(x & (0x80>>i))     // extracting and sending the bits one by one
              DAT = 1;        //  from MSB to LSB
          else
              DAT = 0;
          CLK = 0;    //generate one clock pulse to push the data to the shift register
          CLK = 1;
        }
}
```

```
void delayMs(tWord x)
{
	//delay in terms of milliseconds(approximate)
	// delay(1000) will produce 1 sec delay
	tWord i;
	while(x--)
	  for(i=0;i<75;i++);
}
```

Modification:

- Display RVCE & CSE alternatively.
- Display your name in rolling fashion.

Output & Observations:

<table>
<tr><th rowspan="3">Program No.</th><th colspan="3">Marks for Execution
(7)</th><th colspan="2">Marks for Viva voce
(3)</th><th rowspan="3">TOTAL
(10)</th><th rowspan="3">Signature of the Faculty</th></tr>
<tr><th colspan="3">Rubrics</th><th colspan="2">Rubrics</th></tr>
<tr><th>Understanding of problem
(2)</th><th>Execution
(3)</th><th>Results and Documentation
(2)</th><th>Conceptual Understanding and Communication of Concepts
(2)</th><th>Use of appropriate Design Techniques
(1)</th></tr>
<tr><td>2b</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

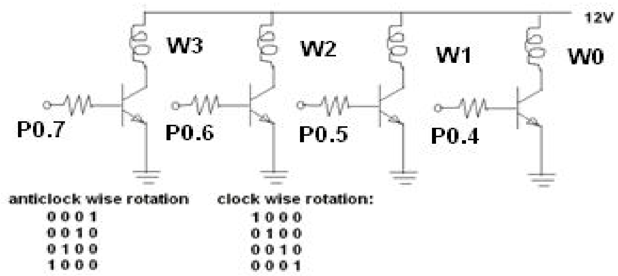
Program No.2b

Paste your DATA SHEET here

[Page intentionally left blank, student to paste his/her data sheet after evaluated by the staff in-charge]

STEPPER MOTOR INTERFACE MODULE

Interfacing Diagram



Stepper Motor Windings - W3-W0
(inside Stepper Motor, leads are provided)
Transistors(SL100) are used to drive windings

- Total number of steps for one revolution = 200 steps (200 teeth shaft)
  Step angle = 360°/200 = 1.8°
- Use appropriate delay in between consequent steps
- 2Phase, 4winding stepper motor is used, along with driver circuit built on the RV All-In-One Card, 12v power is used to drive the stepper motor. Digital input generated by the microcontroller, is used to drive and control the direction and rotation of stepper motors. If it is required to drive bigger/higher torque stepper motors only change is- use MOSFETS or stepper driver ICs to drive motors instead of SL100 transistor.

Experiment No – 3b

AIM: Write an Embedded C program to rotate stepper motor in clock wise and in anti-clock wise direction for “N” steps (Number of steps or angle to rotate to be specified)

Embedded C Program:

```
#include <reg52.h>

typedef unsigned char tByte;
typedef unsigned int  tWord;

//name the windings, P0 bits 7,6,5,4 are connected to stepper windings
sbit W3 = P0^7;
sbit W2 = P0^6;
sbit W1 = P0^5;
sbit W0 = P0^4;

no_of_steps_clk = 100  ; //number of steps to move in clockwise direction
no_of_steps_anticlk = 100 ;//number of steps to move in anti-clockwise direction
void delayMs(tByte);

main()
{   while(1)
    {      W3=1; W2=0; W1=0; W0=0; delayMs(5); if(--no_of_steps_clk==0) break;
           W3=0; W2=1; W1=0; W0=0; delayMs(5); if(--no_of_steps_clk==0) break;
           W3=0; W2=0; W1=1; W0=0; delayMs(5); if(--no_of_steps_clk==0) break;
           W3=0; W2=0; W1=0; W0=1; delayMs(5); if(--no_of_steps_clk==0) break;
    }
    while(1)
    {      W3=0; W2=0; W1=0; W0=1; delayMs(5); if(--no_of_steps_anticlk==0) break;
           W3=0; W2=0; W1=1; W0=0; delayMs(5); if(--no_of_steps_anticlk==0) break;
           W3=0; W2=1; W1=0; W0=0; delayMs(5); if(--no_of_steps_anticlk==0) break;
           W3=1; W2=0; W1=0; W0=0; delayMs(5); if(--no_of_steps_anticlk==0) break;
    }
while(1); //end of program, stay here
}
void delayMs(tByte x)       //delay in terms of milliseconds(approximate)
{                           // delay(1000) will produce 1 sec delay
  tWord i;
  while(x--)
    for(i=0;i<300;i++);
}
```

Modification:

- Rotate the motor by 270^{o} in clock wise direction only and 90^{o} in anti-clock wise direction.
- Rotate the motor by 150 steps.
- Use Timers and rotate the motor for the specified RPM (Revolutions per minute).

Output & Observations:

<table>
<tr><th rowspan="3">Program No.</th><th colspan="3">Marks for Execution
(7)</th><th colspan="2">Marks for Viva voce
(3)</th><th rowspan="3">TOTAL
(10)</th><th rowspan="3">Signature of the Faculty</th></tr>
<tr><th colspan="3">Rubrics</th><th colspan="2">Rubrics</th></tr>
<tr><th>Understanding of problem
(2)</th><th>Execution
(3)</th><th>Results and Documentation
(2)</th><th>Conceptual Understanding and Communication of Concepts
(2)</th><th>Use of appropriate Design Techniques
(1)</th></tr>
<tr><td>3b</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

Program No.3b

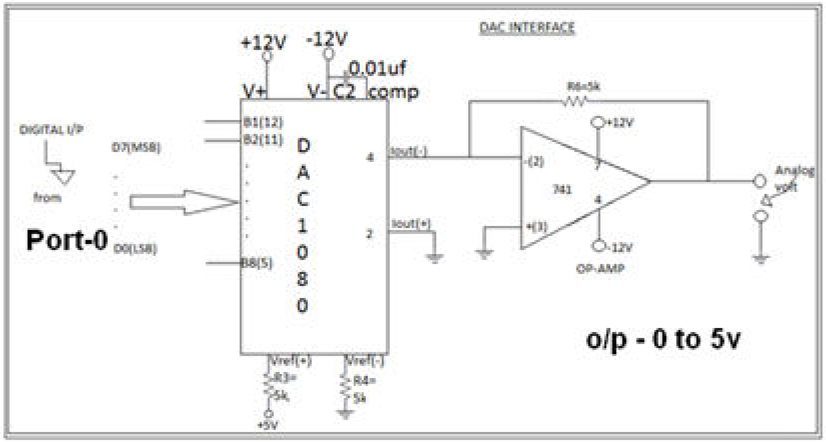
Paste your DATA SHEET here

[Page intentionally left blank, student to paste his/her data sheet after evaluated by the staff in-charge]

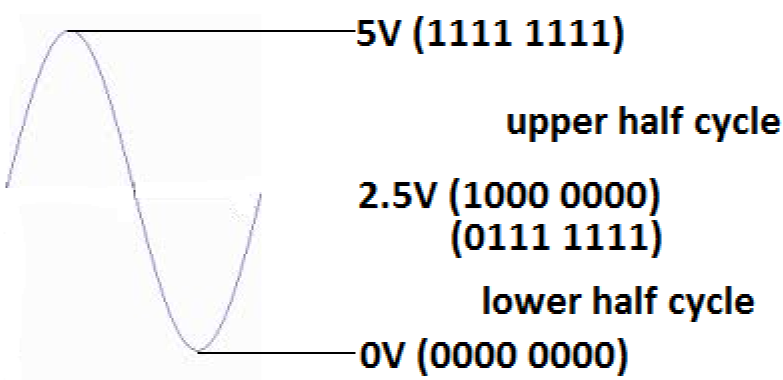
DAC INTERFACE

Description:

DAC refers to Digital to Analog Converter, used to convert digital values to corresponding analog values.



- DAC-080 , an 8 bit Digital to Analog converter IC is used, to convert 8bit Digital I/P to Analog voltage
- Digital I/P : **00 to FF** , corresponding Analog O/P : **0V to 5V**
- Resolution = (5/256) ≈ 20mV



Formula for calculation of the sine table entries: **128 + 127 x Sin ϴ** (128 Corresponds to 80h, i.e. 2.5V, 127 x SIN 90 gives 127, so 128+127 = 255 (for 5v)

Calculate the digital values to be outputted to DAC for angles in the steps of 6^o,

127 x sin 0 = 0	127 x sin 48 = 94
127 x sin 6 = 13	127 x sin 54 = 102
127 x sin 12 = 26	127 x sin 60 = 109
127 x sin 18 = 39	127 x sin 66 = 116
127 x sin 24 = 51	127 x sin 72 = 120

127 x sin 30 = 63 127 x sin 80 = 124
127 x sin 36 = 74 127 x sin 86 = 126
127 x sin 42 = 84 127 x sin 90 = 127

Output the above values in the reverse order to get other portion of the top half cycle,(add 128 for top half cycle, and subtract from 128 for the lower half cycle, refer the table declaration)

Experiment No – 4b

AIM: Write an Embedded C program to generate sine waveform/ half rectified sine waveform/ full rectified sine waveform using DAC module.

Program:

```
#include <reg52.h>

typedef unsigned char tByte;
typedef unsigned int  tWord;

//name the keys located on the RV-USBbased8051 Board

sbit key1 = P3^2;
sbit key2 = P3^3;
sbit key3 = P3^4;

// store the following sine tables in code memory

tByte  code  dac_datas_sine_fullrectified[ ] =
{128+0, 128+13, 128+26, 128+39, 128+51, 128+63, 128+74, 128+84, 128+94, 128+102,
128+109, 128+116, 128+120, 128+124, 128+126, 128+127, 128+126, 128+124, 128+120,
128+116, 128+109, 128+102, 128+94, 128+84, 128+74, 128+63, 128+51, 128+39, 128+26,
128+13};

  // total 30 values

tByte  code  dac_datas_sine_full[ ] =
{128+0, 128+13, 128+26, 128+39, 128+51, 128+63, 128+74, 128+84, 128+94, 128+102,
128+109, 128+116, 128+120, 128+124, 128+126, 128+127, 128+126, 128+124, 128+120,
128+116, 128+109, 128+102, 128+94, 128+84, 128+74, 128+63, 128+51, 128+39, 128+26,
128+13, 128-0, 128-13, 128-26, 128-39, 128-51, 128-63, 128-74, 128-84, 128-94, 128-102,
128-109, 128-116, 128-120, 128-124, 128-126, 128-127, 128-126, 128-124, 128-120, 128-   116,
128-109, 128-102, 128-94, 128-84, 128-74, 128-63, 128-51, 128-39, 128-26,128-13};

 // total 60 values


tByte code  dac_datas_sine_halfrectified [ ] =
{128+0, 128+13, 128+26, 128+39, 128+51, 128+63, 128+74, 128+84, 128+94, 128+102,
128+109, 128+116, 128+120, 128+124, 128+126, 128+127, 128+126, 128+124, 128+120,
128+116, 128+109, 128+102, 128+94, 128+84, 128+74, 128+63, 128+51, 128+39, 128+26,
128+13, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128,
128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128};
 // total 60 values
```

```
 main()
{

tByte i=0,j=0,k=0;
key1=key2=key3=1; //configure as inputs

while(1)
{
        //full rectified sine waveform

        if(key1==0)
          while(1) //continuously output the data in the table to the DAC connected to P0
          {
              P0 = dac_datas_sine_fullrectified[i++];
              if(i==30) i=0;      // total of 30 digital values are stored in the table to
                          // produce full rectified sine wave
              if(key1==0 || key2==0 || key3==0)break;  //check for the key press
          };

        //full sine waveform

        if(key2==0)
          while(1)
          {
              P0 = dac_datas_sine_full[j++];
              if(j==60) j=0;       // total of 30 digital values are stored in the table to
                           // produce full sine wave
              if(key1==0 || key2==0 || key3==0)break;
          }

        //half rectified sine waveform

        if(key3==0)
          while(1)
          {
              P0 = dac_datas_sine_halfrectified[k++];
              if(k==60) k=0;     // total of 30 digital values are stored in the table to
                          // produce half rectified sine wave
              if(key1==0 || key2==0 || key3==0)break;
          };
        }
}
```

Modification:

- Generate Square wave and Triangular waveforms.
- Write Suitable ISR's for handling Key1 and Key2 (INT0 – P3.2 , INT1 – P3.3) in displaying the Square wave and Triangular waveforms.

Output & Observations:

Program No.	Marks for Execution (7)			Marks for Viva voce (3)		TOTAL (10)	Signature of the Faculty
	Rubrics			Rubrics			
	Understanding of problem (2)	Execution (3)	Results and Documentation (2)	Conceptual Understanding and Communication of Concepts (2)	Use of appropriate Design Techniques (1)		
4b							

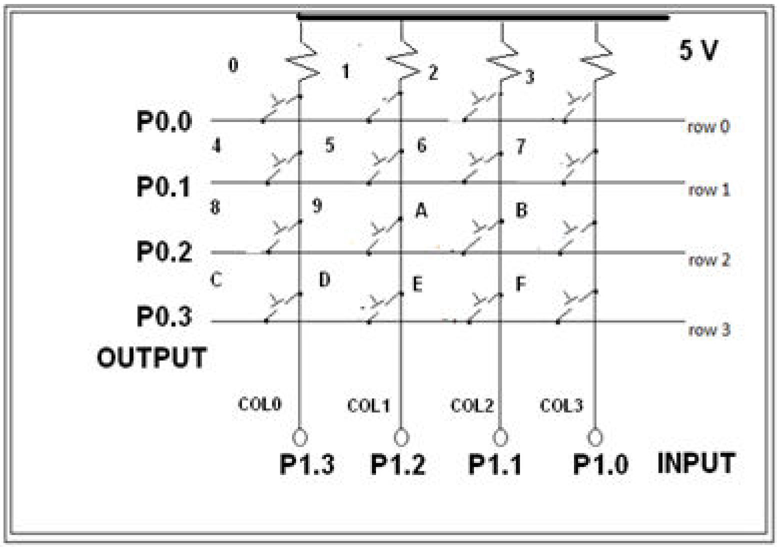
Program No.4b

Paste your DATA SHEET here

[Page intentionally left blank, student to paste his/her data sheet after evaluated by the staff in-charge]

MATRIX KEYBOARD INTERFACE

Description:



- If no key is pressed, we will have on columns 0-3, '1111' on P1.3 to P1.0, as all the inputs are pulled up by pull up resistors.
- If we press any key, let '0' key be pressed, it will short row0 and col0 lines (P0.0 & P1.3), so whatever data (0 or 1) available at row0 (P0.0) is available at col0 (P1.3). Since already columns are pulled high, it is required to apply logic '0' to see change in col0 when the key is pressed.
- To identify which key is pressed,
  - Check for a key press in first row by out putting – '0111'on row's, check which column data is changed, if no key press go for next row
  - Check for a key press in second row by out putting – '1011'on row's, check which column data is changed, if no key press go for next row
  - Check for a key press in third row by out putting – '1101'on row's, check which column data is changed, if no key press go for next row
  - Check for a key press in last row by out putting – '1110'on row's, if no key is pressed go for the first row again
- Once the key press is found, use the row number and column number and look up table to convert the key position corresponding to ascii code. Use appropriate delay for debouncing.

Experiment No – 5b(i)

AIM: Write an Embedded C program to interface 4 X 4 matrix keyboard using lookup table and display the key pressed on the Monitor.

Program:

```
#include <reg52.h>

typedef unsigned char tByte;
typedef unsigned int  tWord;

// name the rows and columns of 4 x 4 keyboard
sbit row0 = P0^0;
sbit row1 = P0^1;
sbit row2 = P0^2;
sbit row3 = P0^3;

sbit col0 = P1^3;
sbit col1 = P1^2;
sbit col2 = P1^1;
sbit col3 = P1^0;

// key look up table, containing key codes
tByte code keys[4][4] =   {     {'0','1','2','3'},
                                {'4','5','6','7'},
                                {'8','9','a','b'},
                                {'c','d','e','f'}     };
void delayMs(tByte x);

main()
{
tByte row_pos, col_pos;

row0=row1=row2=row3=0;        //as outputs
col0=col1=col2=col3=1;        //as inputs

//configure the serial port & the timer1 used for 9600 baud generation
SCON = 0x50; TMOD = 0X20; TH1 = -3; TR1 = 1; TI = 1;
while(1)
{       while(1)
        { //select the first row  & check for key press in row0
          row0=0; row1=1; row2=1; row3=1; row_pos=0;
          if(col0==0){col_pos=0;break;}
          if(col1==0){col_pos=1;break;}
          if(col2==0){col_pos=2;break;}
          if(col3==0){col_pos=3;break;}
```

```
            //select the second row  & check for key press in row1

            row0=1;row1=0;row2=1;row3=1; row_pos=1;
          if(col0==0){col_pos=0;break;}
          if(col1==0){col_pos=1;break;}
          if(col2==0){col_pos=2;break;}
          if(col3==0){col_pos=3;break;}

            //select the third row  & check for key press in row2
            row0=1;row1=1;row2=0;row3=1; row_pos=2;
          if(col0==0){col_pos=0;break;}
          if(col1==0){col_pos=1;break;}
          if(col2==0){col_pos=2;break;}
          if(col3==0){col_pos=3;break;}

            //select the fourth row  & check for key press in row3
            row0=1;row1=1;row2=1;row3=0; row_pos=3;
          if(col0==0){col_pos=0;break;}
          if(col1==0){col_pos=1;break;}
          if(col2==0){col_pos=2;break;}
          if(col3==0){col_pos=3;break;}
        }
        delayMs(20); //debounce

         /* use the following lines if you want to output the key code on P0 for a second
         P0 = keys[row_pos][col_pos];     // output the keycode on P0 for 1sec
         delayMs(1000);  */

         SBUF = keys[row_pos][col_pos];    // output the keycode on serial port, (terminal)

        while(col0==0 || col1==0 || col2==0 || col3==0);  //wait for the key release
        delayMs(20);                  //debounce
    }
}

void delayMs(tWord x)  //delay in terms of milliseconds(approximate)
{                            // delay(1000) will produce 1 sec delay
 tWord i;
 while(x--)
   for(i=0;i<300;i++);
}
```

Modification:

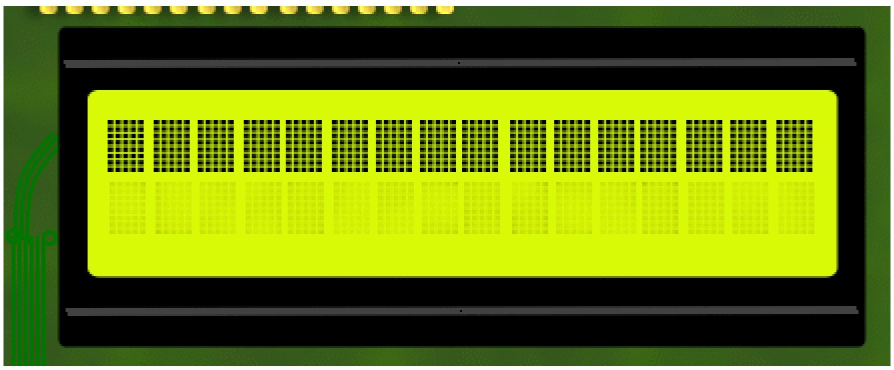
- Display the Key code of the key pressed on 7 segment display device

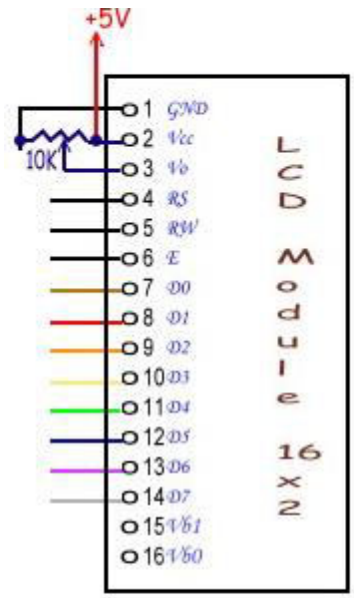
Output & Observations:

LCD INTERFACE

Description:

LCD’s are preferred to seven segment displays because of their versatility and capability to house more information. 2 line (16x2) is the most popular, low cost character oriented LCD, suitable for understanding the working and programming of LCD. You have seen LCD modules used in many of the electronics devices like coin phone, billing machine and weighing machines. It is a powerful display options for stand-alone systems. Because of low power dissipation, high readability, flexibility for programmers, LCD modules are becoming popular.





LCD consists of DDRAM, CGROM, Shift registers, bit/pixel drivers, refreshing logics and lcd controller. The data to be displayed on lcd, is to be written on to the DDRAM-display data Ram using the ascii format. CGROM-Character generator rom, contains dot/pixel patterns for every character to be displayed (pre programmed). Shift registers are used to convert CGROM parallel data to serial data(serializing), drivers are required to drive (ON/OFF) the bits, refreshing logics are required to hold the display data, as the dots are displayed row by row basis continuously, like in CRT.

LCD provides many control pins, to enable the microcontroller or microprocessor to communicate, whatever the data we write to LCD is of two types, either it is a command to the LCD(to configure) or ASCII code of character to be displayed on LCD (to DDRAM). RS signal is used for this,

RS - 0, writing command byte into command register of LCD
1, writing data (ASCII code) into Data register of LCD
R/W - 0, Write to LCD (Data/Command)
1, Read from the LCD
E - Enable is required to perform the writing/reading to LCD,

E – ‘1’ (for 450nsec) & then ‘0’ (High to Low Pulse)
D0-D7 - It is a bidirectional data bus, used to write data/command to LCD or reading status.

Instruction	D7	D6	D5	D4	D3	D2	D1	D0	Description
Clear display	0	0	0	0	0	0	0	1	Clears Display and returns cursor to home position.
Cursor home	0	0	0	0	0	0	1	X	Returns cursor to home position. Also returns display being shifted to the original position.
Entry mode set	0	0	0	0	0	1	I/D	S	I/D = 0 → cursor is in decrement position. I/D = 1 → cursor is in increment position. S = 0 → Shift is invisible. S = 1 → Shift is visible
Display ON- OFF Control	0	0	0	0	1	D	C	B	D- Display, C- Cursor, B-Blinking cursor 0 → OFF 1 → ON
Cursor/ Display Shift	0	0	0	1	S/C	R/L	X	X	S/C = 0 → Move cursor. S/C = 1 → Shift display. R/L = 0 → Shift left. R/L = 1→ Shift right.
Function Set	0	0	1	DL	N	F	X	X	DL = 0 → 4 bit interface. DL = 1 → 8 bit interface. N = 0 → 1/8 or 1/11 Duty (1 line). N = 1 → 1/16 Duty (2 lines). F = 0 → 5x7 dots. F = 1 → 5x10 dots.

Programming LCD

Two steps are involved,

1.Configure the LCD for different parameters/settings, by writing series of commands (command bytes) like

- Function set command(0x38)
- Display On command(0x0C)
- Clear display (0x01)

2. Writing actual string data to LCD, character by character, (by default characters are displayed from line1 first column position, we can issue DDRAM address command - 0x80 + char pos, for first line, 0xc0 + char pos, for second line).

5b (ii) AIM: Write an Embedded C program to display the strings, on 2x16 character LCD.

Embedded C Program:

```
#include <reg52.h>
#include <intrins.h>

typedef unsigned char tByte;
typedef unsigned int  tWord;

//name the LCD pins
sbit RS   = P1^4;      // 0 - command   1 - data
sbit RW = P1^5;       // 0 - write     1 - read
sbit E     = P1^6;      // 1 to 0, performs writing of command/data

#define LCDData P0

// function prototypes
void LCD_DispStr(tByte line_no,char* str);
void LCD_Init(void);
void LCD_Command(tByte command);
void LCD_Data(tByte databyte);
void enpulse(void);
void delay(tByte val);

main()
{
tByte str1[] = "Hello..RVCE..";
tByte str2[] = ".....CSE.....";

//configure the pins
RS=RW=E = 0;    // as output
LCDData = 0;       // as output

LCD_Init();
LCD_DispStr(1,str1);
LCD_DispStr(2,str2);

while(1);         // stay here indefinitely
}

void LCD_DispStr(tByte line_no,char* str)
{
        tByte i=0;
        if(line_no==1)
```

```
      LCD_Command(0x80);    // command to set the memory ptr to first line
    else
      LCD_Command(0xc0);// cmd to set the mem ptr to first char of second line

    while(str[i]!='\0')
    {
      LCD_Data(str[i]);i++;
      if(i==16)break;       // as max of 16 chars per line
    }
}
void LCD_Init(void)
{
    LCD_Command(0x38);      //function set- 2 line display,byte mode
    LCD_Command(0x0c);      //display on
    LCD_Command(0x01);      //clear the display
}
void LCD_Command(tByte command) //to send command to the lcd
{
    RS=0;
    RW=0;
    LCDData = command;
    enpulse();              // generate enable pulse
    delay(50);
}
void LCD_Data(tByte databyte)       //to send data to the lcd
{
    RS=1;                   //data is written
    RW=0;
    LCDData = databyte;
    enpulse();
    delay(50);
}
void enpulse(void)                  // to generate enable pulse 1 to 0, on Enable pin
{
    E=1;
    delay(2);
    E=0;
    delay(2);
}
void delay(tByte val)
{   tByte i;
    for(i=0;i<val;i++)
    { _nop_();
      _nop_();
      _nop_();
      _nop_();
      _nop_();
    }
}
```

Program No.	Marks for Execution (7)			Marks for Viva voce (3)		TOTAL (10)	Signature of the Faculty
	Rubrics			Rubrics			
	Understanding of problem (2)	Execution (3)	Results and Documentation (2)	Conceptual Understanding and Communication of Concepts (2)	Use of appropriate Design Techniques (1)		
5b (i)							
5b (ii)							

Program No.5b

Paste your DATA SHEET here

[Page intentionally left blank, student to paste his/her data sheet after evaluated by the staff in-charge]

Experiment 6a:

AIM: Write and simulate ARM assembly language programs.

1. Program to add 3 numbers stored in ROM and store the answer in R/W memory.

Program:

```
AREA  RESET  ,  CODE
        ENTRY

         MOV   R3, #0
         MOV   R4, #0
         LDR   R0, =INPUT
         LDR   R1, =OUTPUTS
         LDR   R2, [R0]
         ADD   R4, R4, R2
         ADD   R0, R0, #4
         ADD   R3, R3, #1
         CMP   R3, #3
         BNE   CONT
         STR   R4, [R1]
         STOP  B   STOP

INPUTS      DCD    01,02,03
AREA        MEMORY, DATA
OUTPUTS   SPACE   4
END
```

<u>Output & Observations:</u>

2. Program to find the Sum of 3x + 4y + 9z, where x = 2, y=3 and z=4.

```
AREA   RESET , CODE

     MOV R1,  #2   ; Let x = 2
     MOV R2,  #3   ; Let y = 3
     MOV R3,  #4   ; Let z = 4

     ADD  R1, R1, R1, LSL #1
     MOV R2, R2, LSL #2
     ADD  R3, R3, R3, LSL #3
     ADD  R1, R1, R2
     ADD  R1, R1, R3
STOP       B   STOP
END
```

<u>Output & Observations:</u>

3. Write an ARM program to perform division of 500 by 16 using repeated subtraction.

```
AREA   DIV, CODE
ENTRY

         MOV     R1, #500
         MOV     R2, #16
         MOV     R3, #0
         MOV     R4, R1
REPT     SUBS    R4, R4, R2
         ADDPL   R3, R3, #1
         BPL     REPT
         ADDMI   R4, R4, R2
         STOP  B STOP
END
```

Output & Observations:

4. Write a program to calculate $3x^2 + 5y^2$ where x=8 and y=5

```
AREA PROCED, CODE
ENTRY

         MOV R2, #8
         BL SQUARE
         ADD R1, R3, R3, LSL #1
         MOV R2, #5
         BL SQUARE
         ADD R4, R1, R0
         STOPB STOP

SQUARE MUL R3, R2, R2
         MOV PC, LR
END
```

Output & Observations:

Program No.	Marks for Execution (7)			Marks for Viva voce (3)		TOTAL (10)	Signature of the Faculty
	Rubrics			Rubrics			
	Understanding of problem (2)	Execution (3)	Results and Documentation (2)	Conceptual Understanding and Communication of Concepts (2)	Use of appropriate Design Techniques (1)		
6a							

Program No.6a

Paste your DATA SHEET here

[Page intentionally left blank, student to paste his/her data sheet after evaluated by the staff in-charge]

Experiment No – 6b

ARM Interfacing Experiments
[Demonstrate the output using Keil simulator, select Device as NXP's LPC 2148]

1. **Generate the asymmetric square wave on the P0.1 pin using software**

Embedded C Program: (Create the project in Keil using LPC 2148)

```
#include <LPC214X.H>
int main(void)
      {
        unsigned int x;
        IODIR0 = 0xFFFFFFFF;    //Make all the pins as outputs
        for(;;)
           {
                IOSET0 = 1 << 10;  //Set the port pin P0.10
                for(x=0;x<30000;x++);  //delay for ON time
                IOCLR0 = 1 << 10; //Clear the port pin P0.10
                      for(x=0;x<40000;x++);  // delay for OFF time
             }
        }
```

Output & Observations:

2. **Generate the square wave of frequency 1KHz using the timer, on P0.10pin**

Steps-

1. Load a number in the match register,
   Let us assume PCLK = 15 MHz (CCLK -60MHz,%by 4 using VPB register setting),
   count = Time period of required output(Td) / time period of input frequency(T),
   Td = 1/1KHz = 1 msec, half of it is 0.5msec; T = 1/15MHz = 0.067µsec
   = 0.5 msec/0.067µsec = 7462
2. Load the MCR for stopping the timer on match & disable the interrupt
3. Start the timer, by enabling the 'E' bit in TCR
4. Now TC starts counting, when it matches with the MR value, it stops counting
5. Stop the timer

Embedded C Program:

```
#include  <LPC214x.h>
void delay(void);
```

```
int main(void)
     {
       T0MR0 =  7462;  //use the Timer0 and load the MR0 with count
        T0MCR = 0X0004;  //   0000….100 – Stop the timer, after match
        IODIR0 |=  (1<<10);    // set the direction as output, without disturbing other bits
       While(1)
       {
              I0SET0  =  1 << 10;  //set P0.10 to 1
          delay();
              I0CLR0  =  1 <<10;  //clear P0.10 to 1
              delay();
        }
   }
void   delay(void)
     {
          T0TCR = 1;  //start the timer
          While (!(T0TC == T0MR0));
          T0TCR = 2; // reset the counter  and stop the timer
          T0TC = 0;
     }
```

Output & Observations:

3. Assignment : Generate 25% duty cycle wave form on pin P0.2, using PWM of LPC 2148

Write the Code & Output :

Mini Project

AIM: Design, Interface and Develop Embedded C program to Build Temperature controlled Fan. [Use Temperature sensor, ADC 0804 / inbuilt Analog channel, DC/AC Motor]. Develop suitable Windows/Linux based application to display the temperature received from the Microcontroller kit.
(Any programming language can be used)

Components Required:

Schematic Circuit:

Output & Observations:

[Enclose the photo of the working model]

<table>
<tr><th rowspan="3">Program No.</th><th colspan="3">Marks for Execution
(7)</th><th colspan="2">Marks for Viva voce
(3)</th><th rowspan="3">TOTAL
(10)</th><th rowspan="3">Signature of the Faculty</th></tr>
<tr><th colspan="3">Rubrics</th><th colspan="2">Rubrics</th></tr>
<tr><th>Understanding of problem
(2)</th><th>Execution
(3)</th><th>Results and Documentation
(2)</th><th>Conceptual Understanding and Communication of Concepts
(2)</th><th>Use of appropriate Design Techniques
(1)</th></tr>
<tr><td>9</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

Program – Mini Project

Paste your DATA SHEET here

[Page intentionally left blank, student to paste his/her data sheet after evaluated by the staff in-charge]

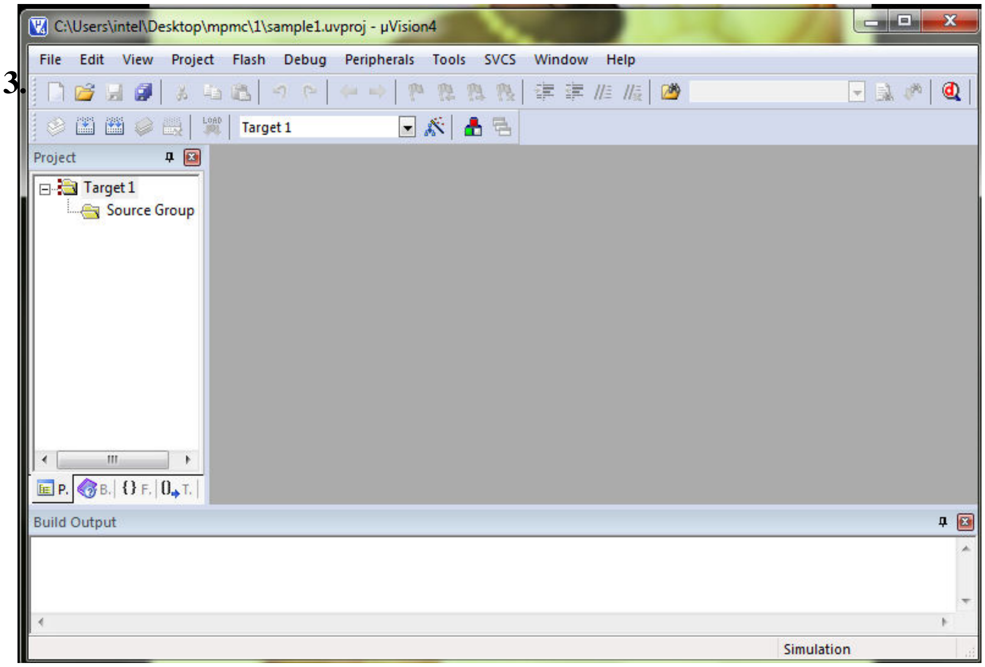
STEPS to generate Target HEX file using Keil.

1. **Create a Folder with a name MC in the Desktop.**

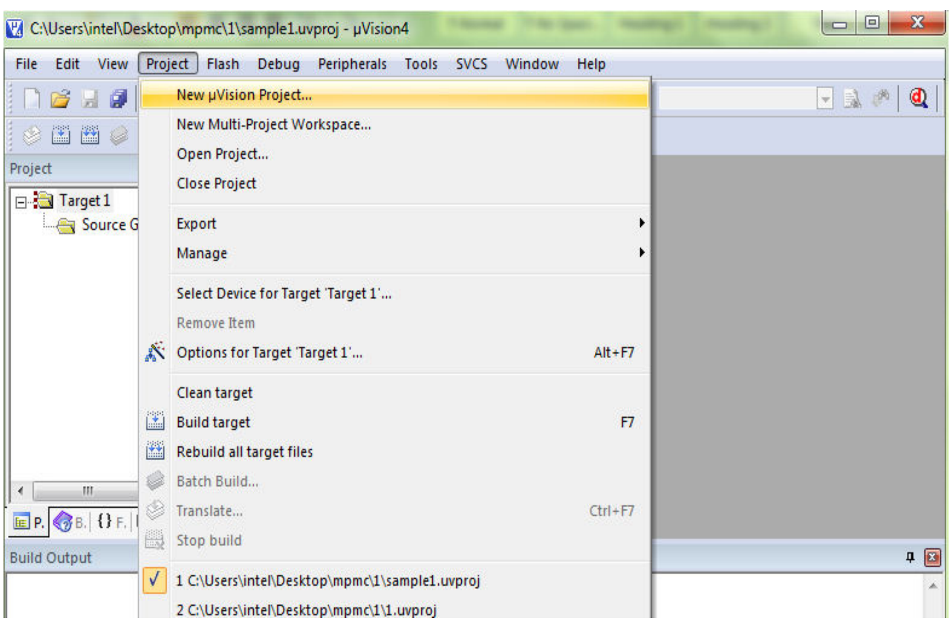
2. **Click on Icon** 

, this opens the KEIL window shown below….

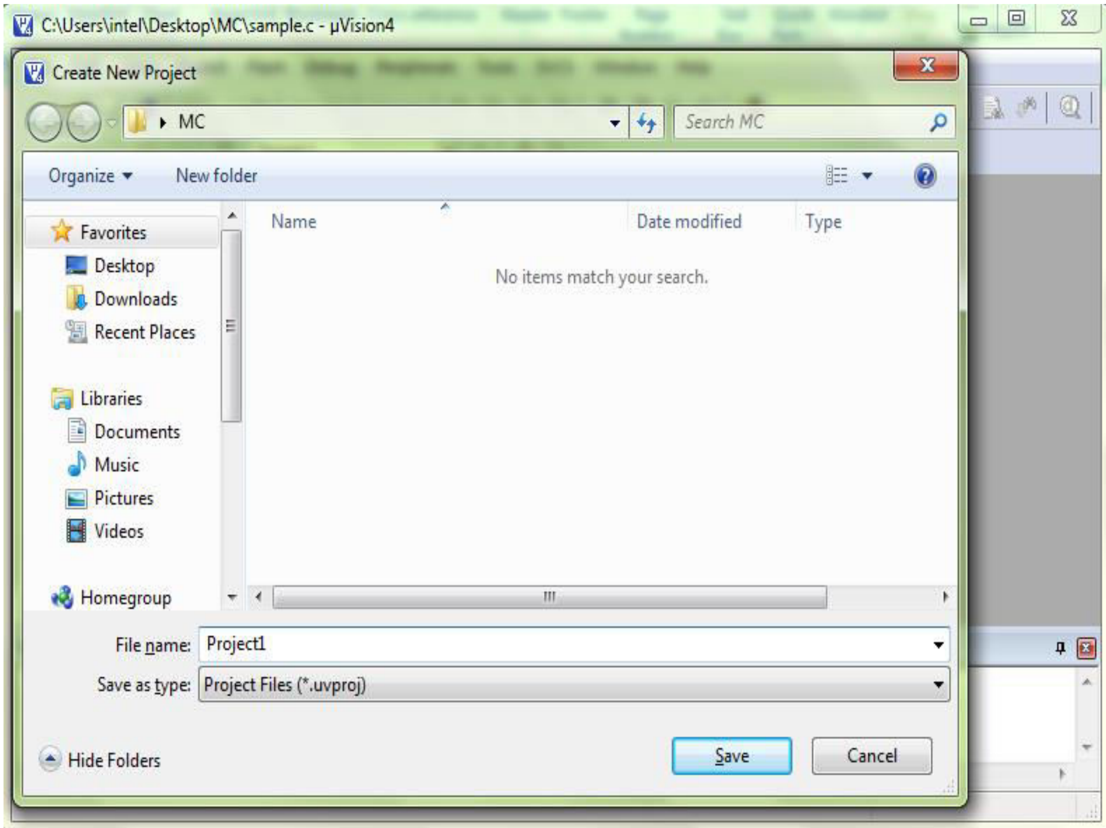
3.



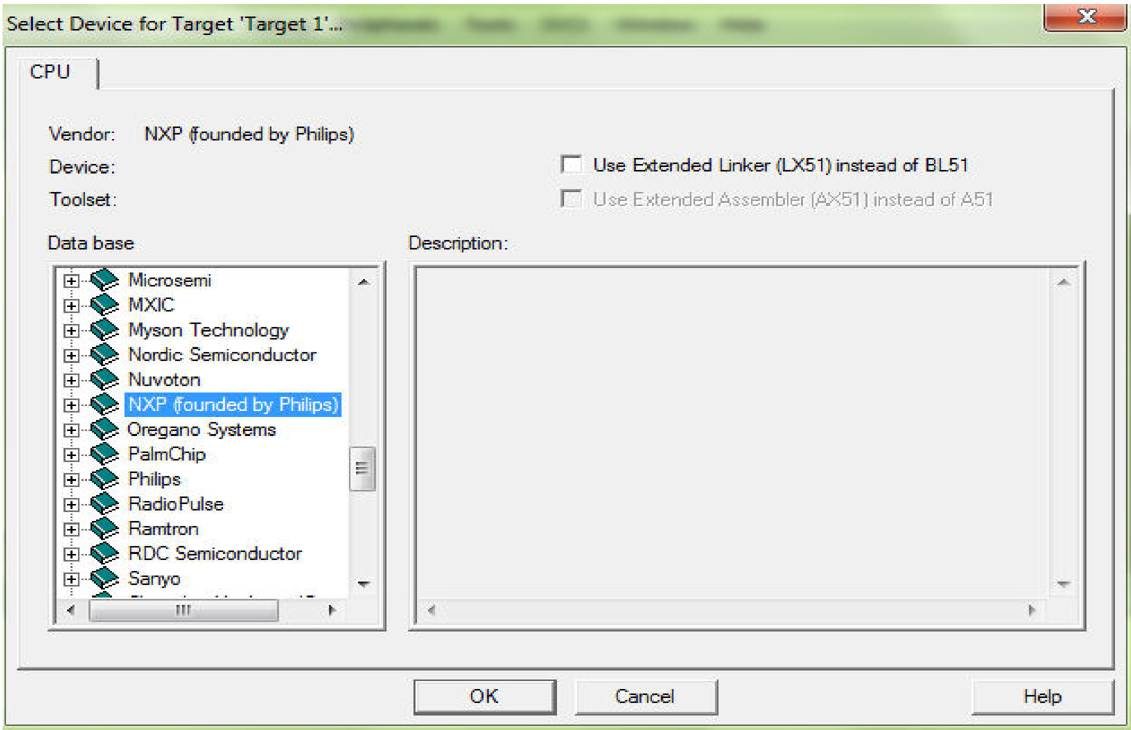
1. **From the Menu bar select Project →New µVision Project**

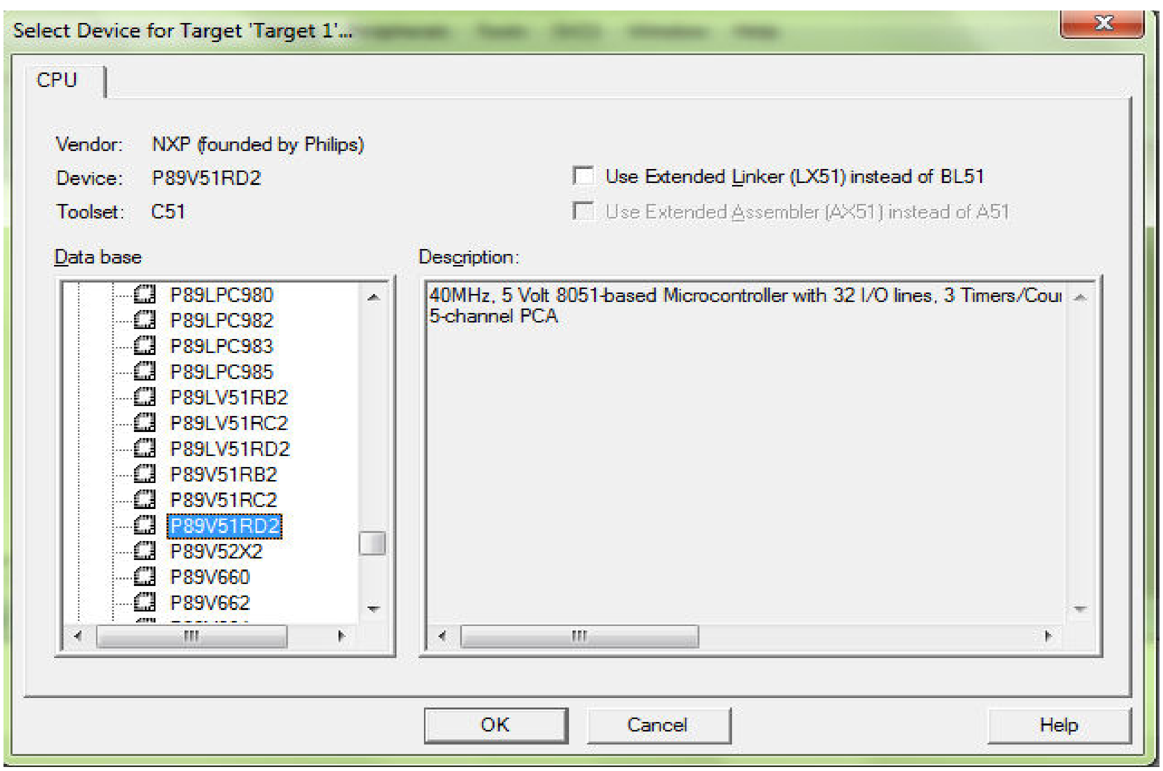


4. **Give a name to the project – Project1 and save in the MC folder in the Desktop.**

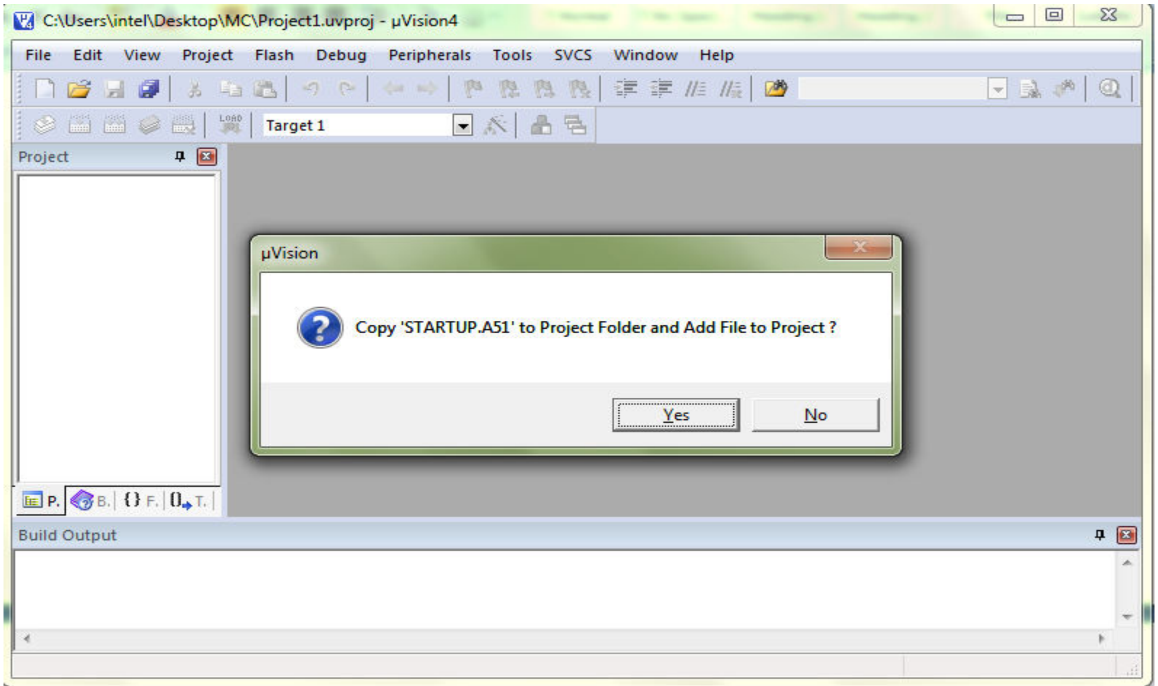


5. **Select Device for the target → NXP (FOUNDED BY Philips) followed with the Microcontroller selection → P89V51RD2 and then click on OK.**

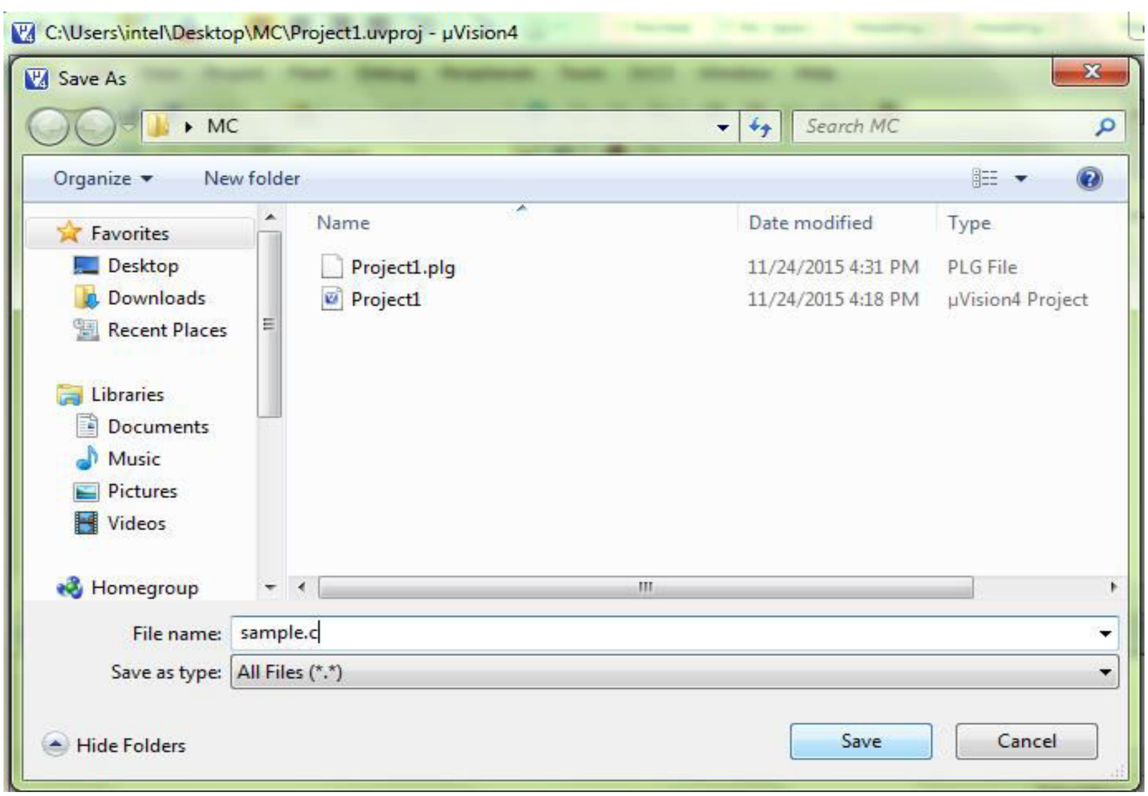




6. **A pop up window appears, asking for Startup files to be included for the project . Click on NO**

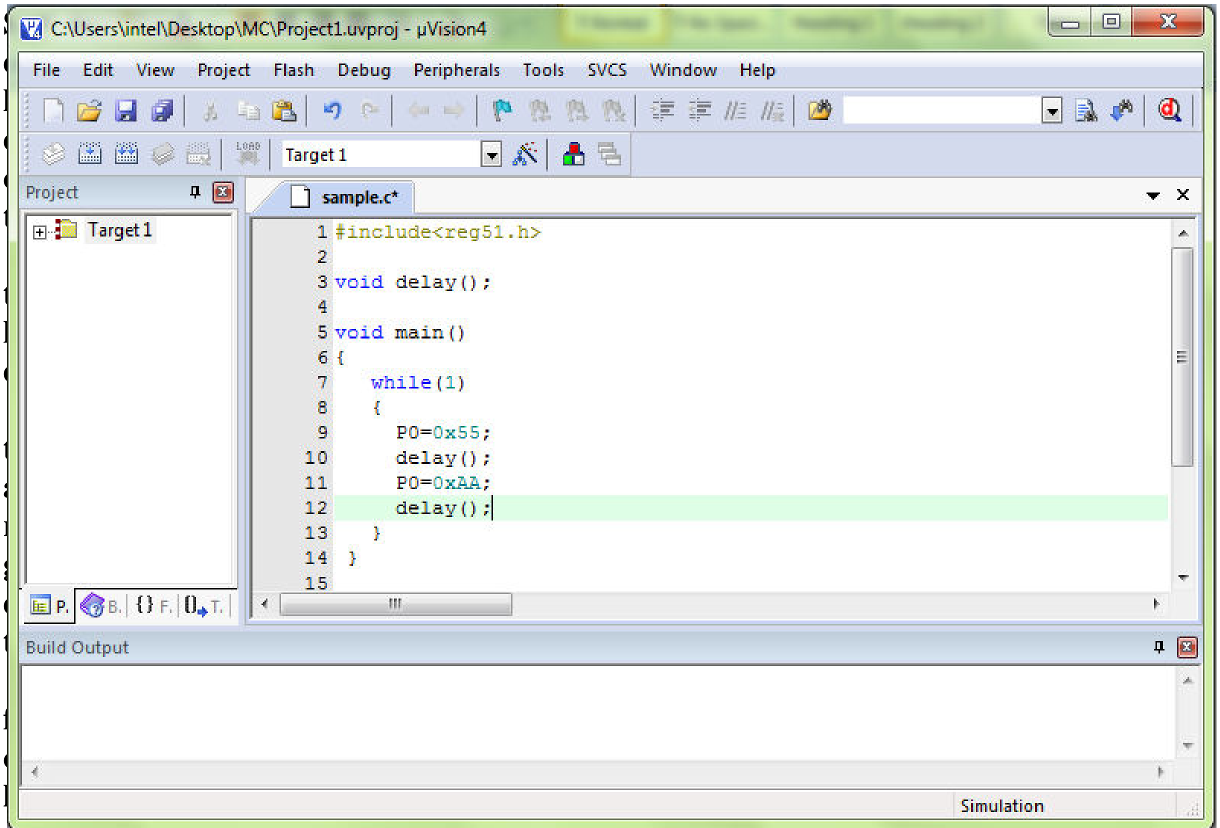


7. **In the Menu bar Select File →New , create a new file and save the file with the extension .c, for an embedded C file , for example sample.c. For an ALP file the extension is .asm.**



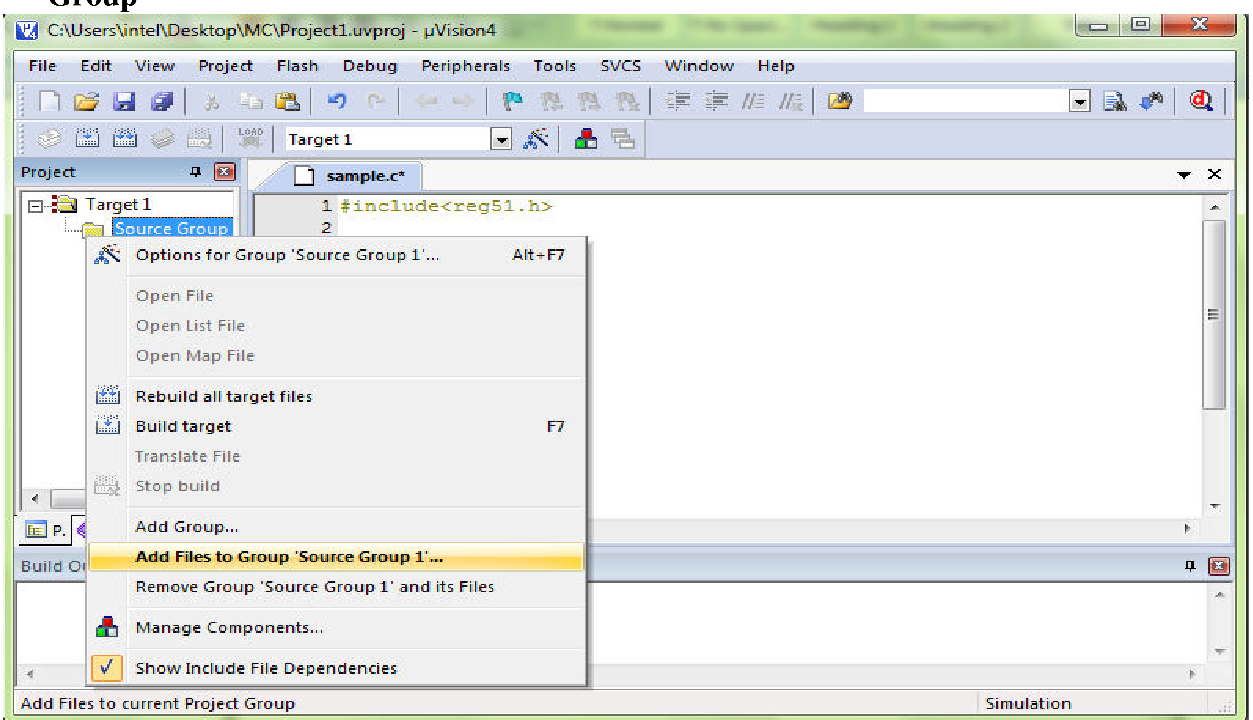
8. Type the embedded c program

9.

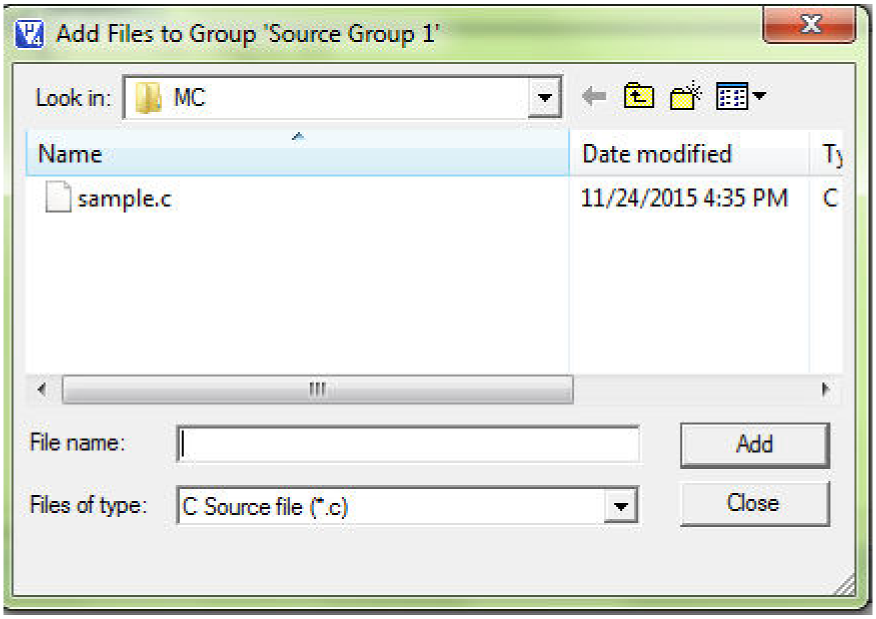


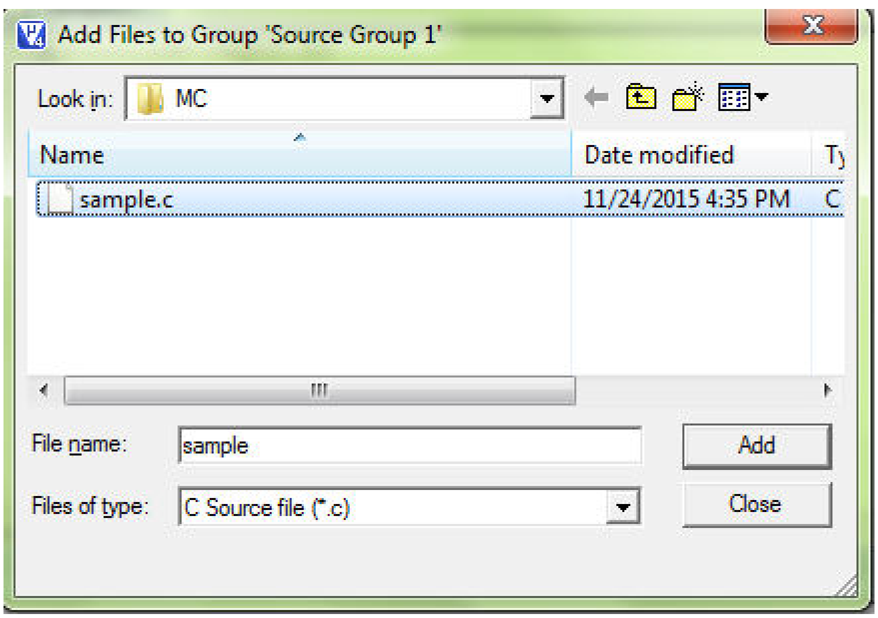
d

er towards the left window , then right click on Source Group →Add files to Group

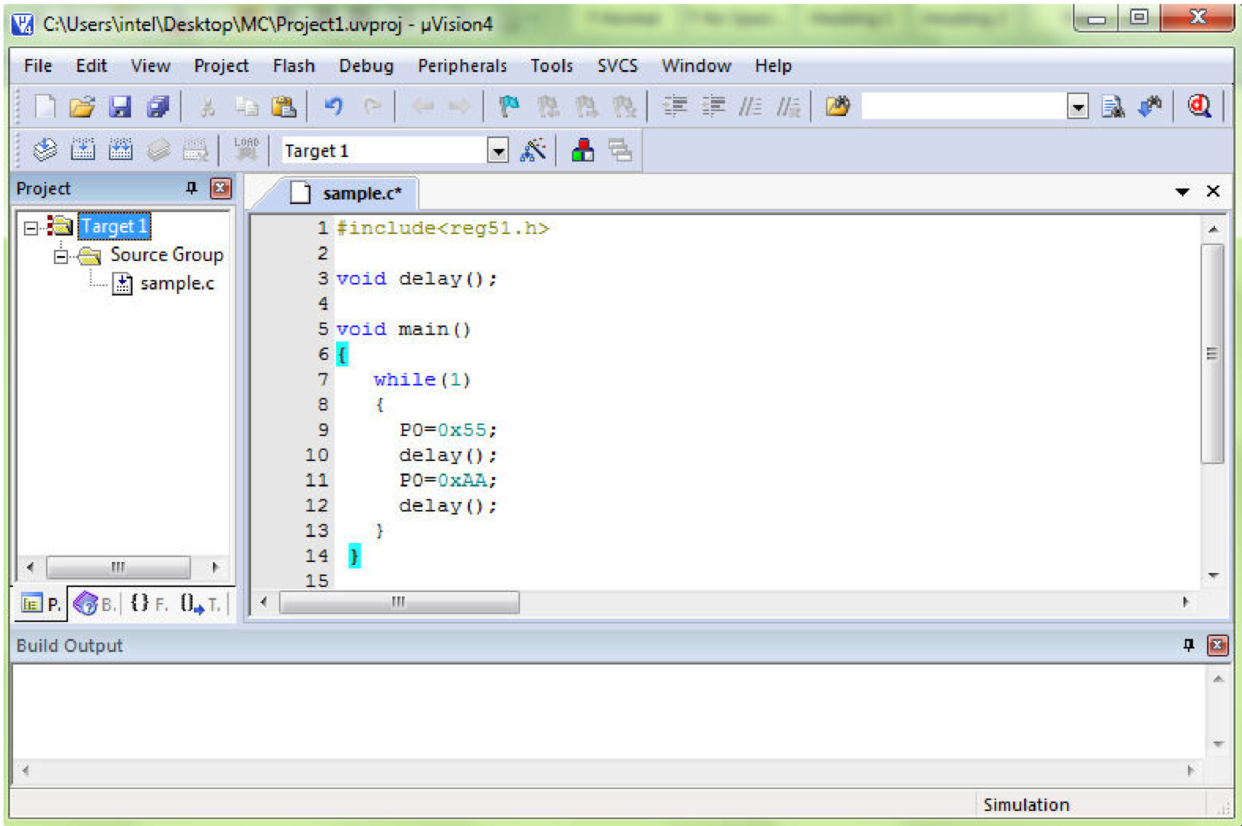


10. The below Pop up window appears , select your file i.e, sample.c , click on Add , then Close.



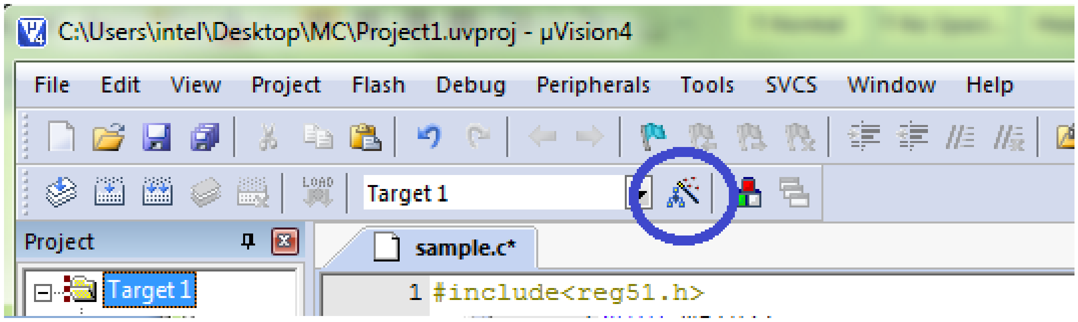


11. Click on Source Group , you can find your file added .



12. Ensure Source Group contain only your file , if others exist right click on the other files and remove it.

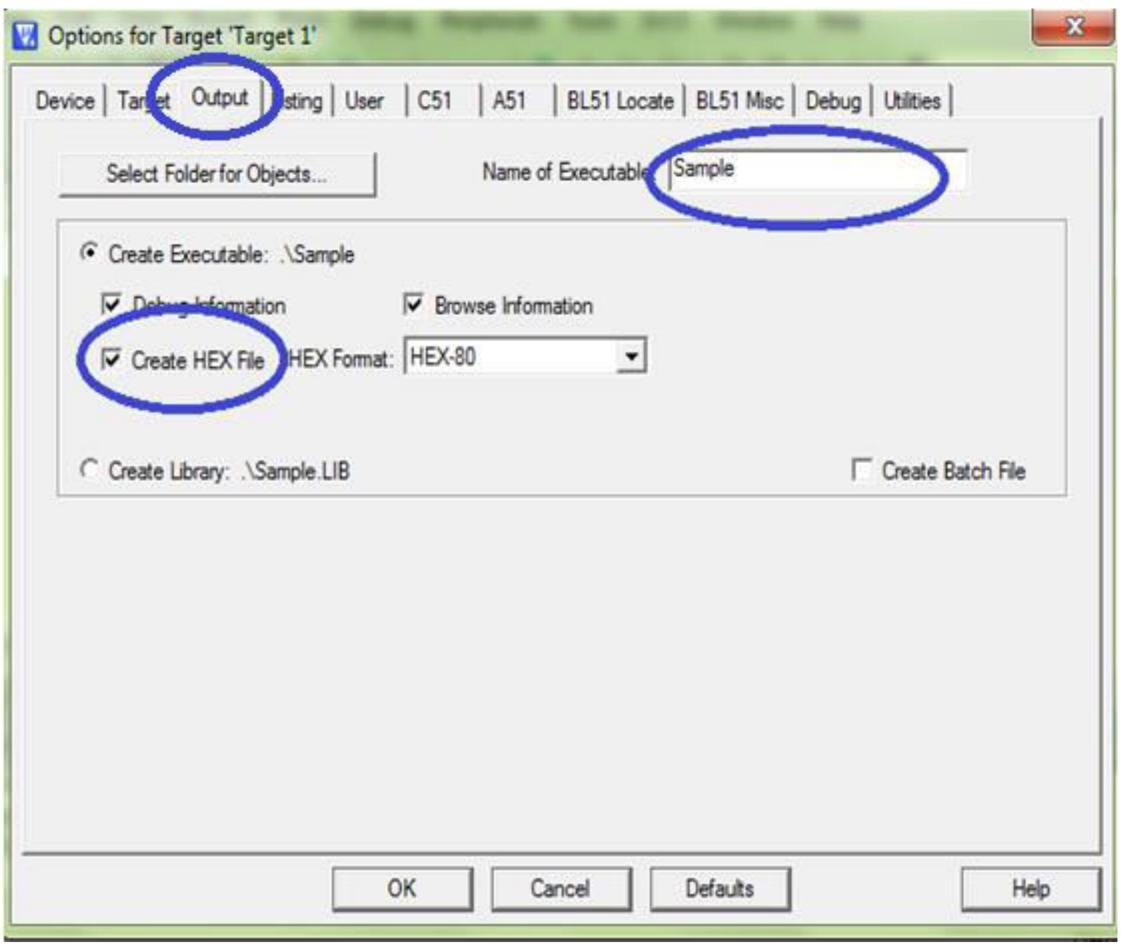
13. Click on options button, if not present then select it from Menu bar, i.e, Project → options



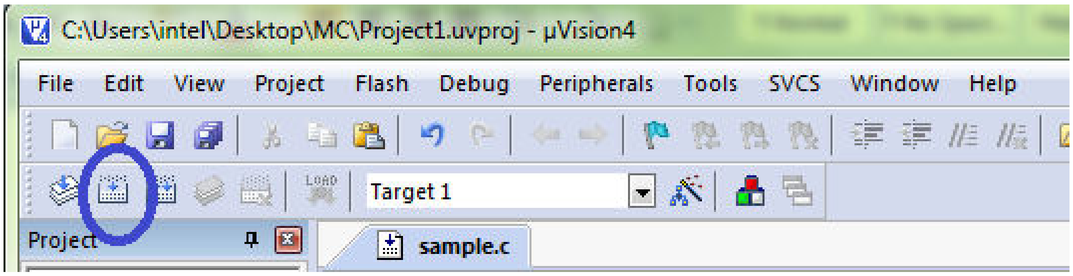
14. In the options POP UP window select the OUTPUT

a. Tick on Create HEX File,

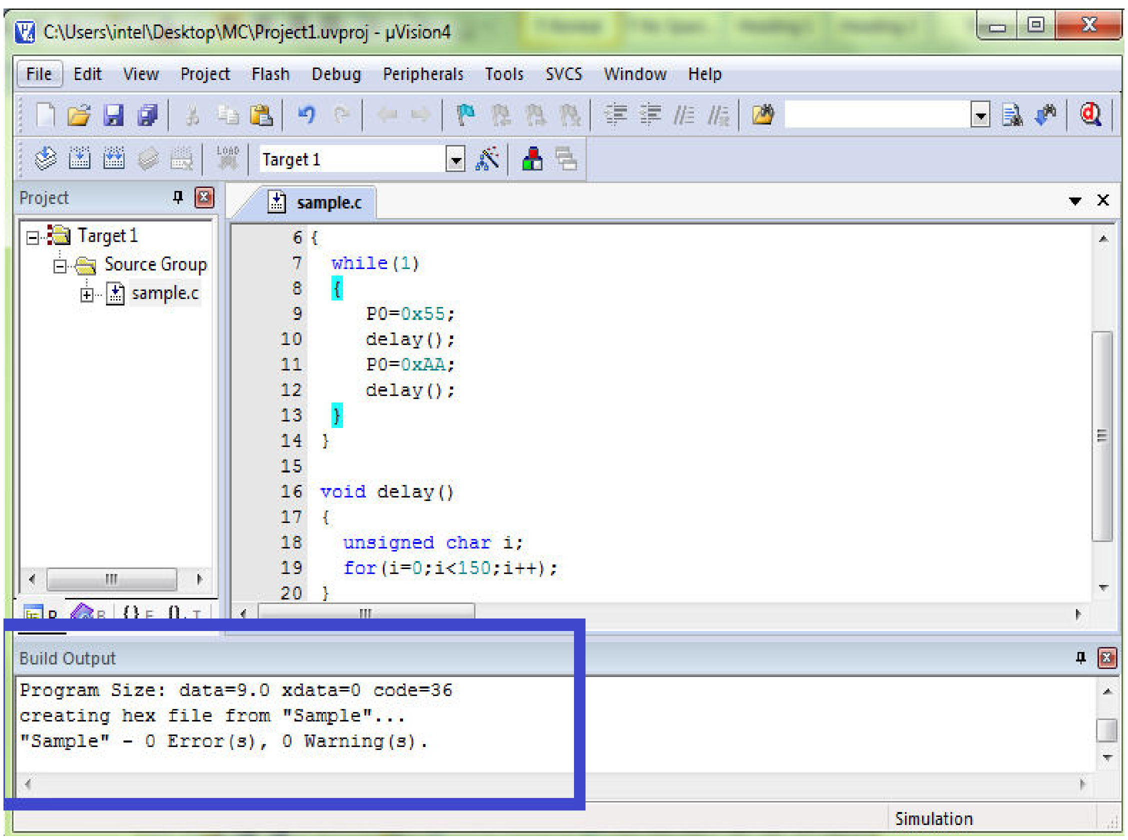
b. Name of the Executable program would be the project name, if required you can modify, and click on OK.



15. Click on the Build Button or select it from Menu Bar , Project → Build Target



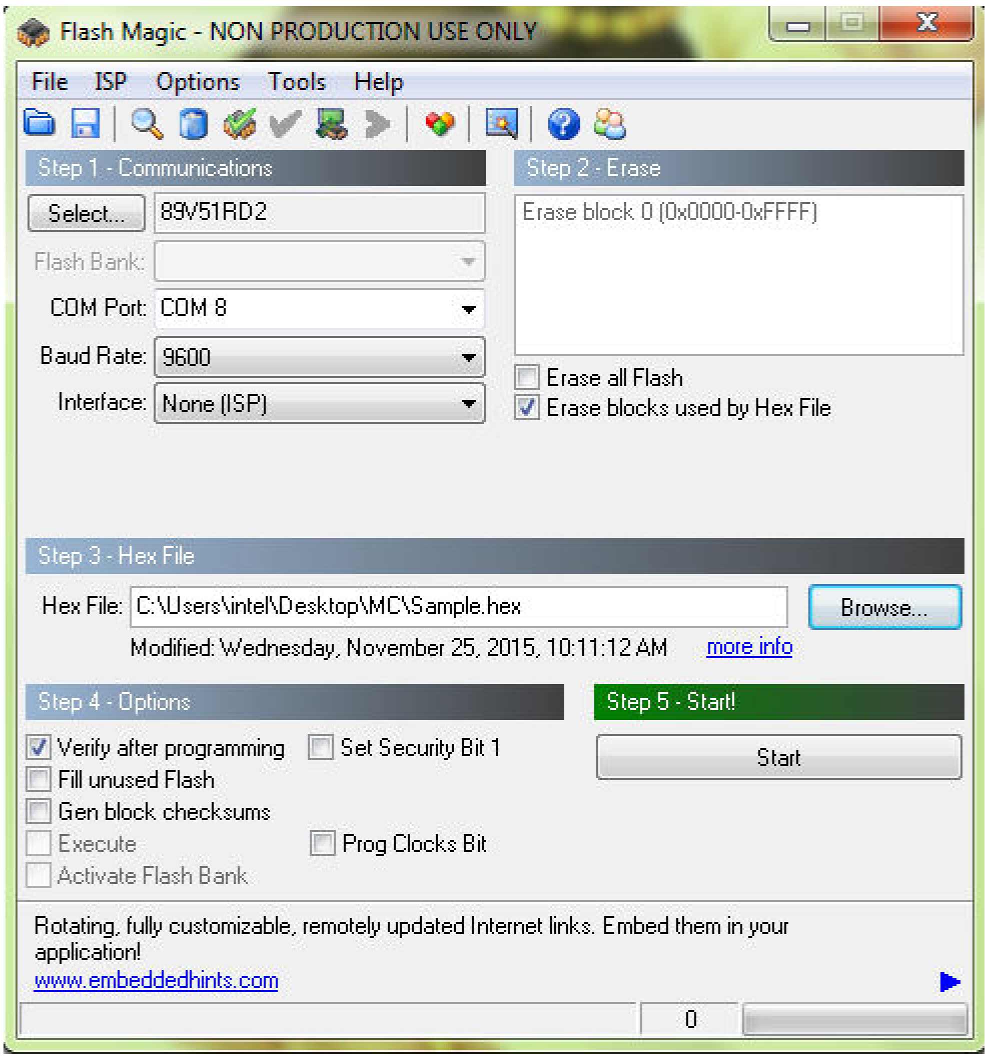
16. This creates the Target HEX file, if there are no syntax errors in the program. Otherwise it list the errors , debug the errors save your file and build the target again.



17. Next we have to load the HEX file on to the MICROCONTROLLER, for which Flash Magic is used.

18. On the Desktop click on the ICON 

which opens the window shown below.



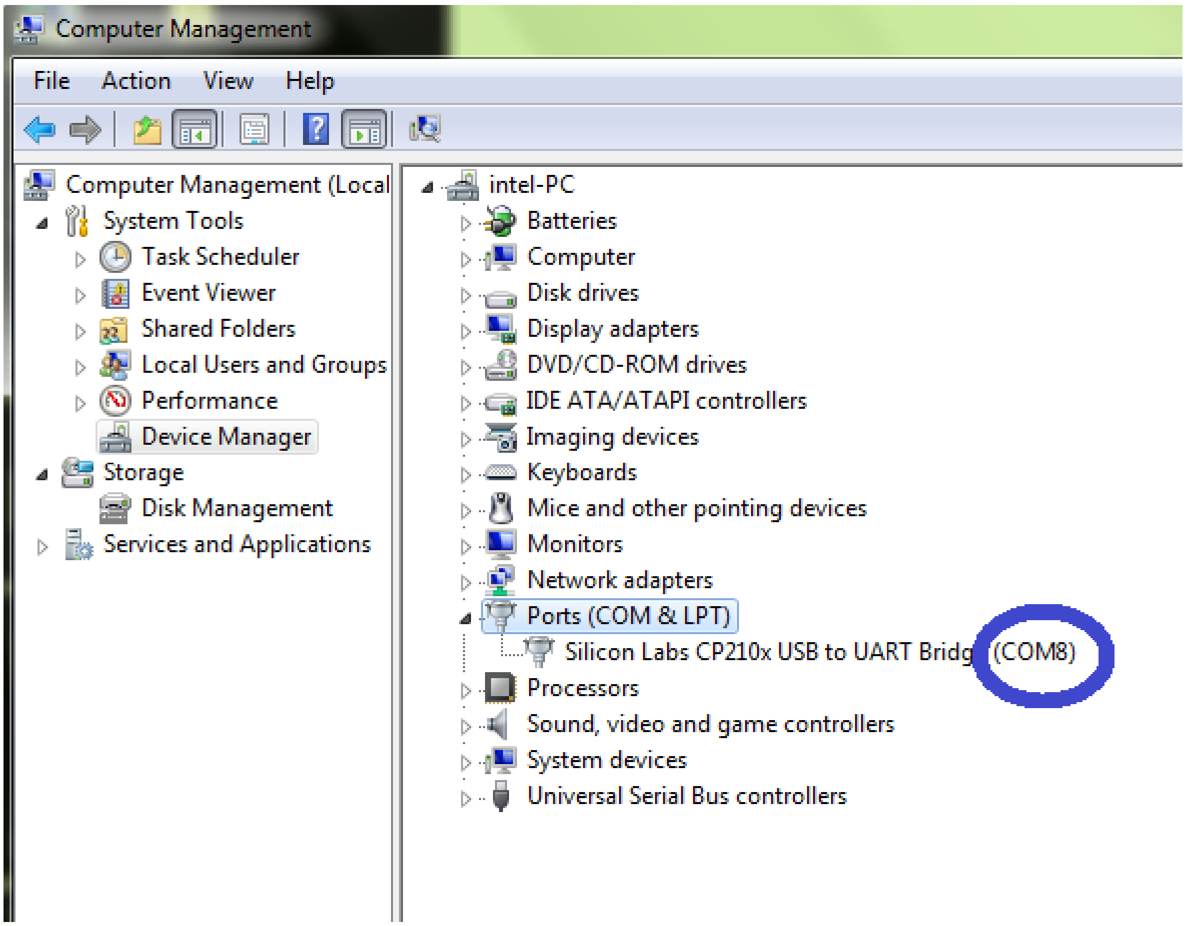
19. In Step 1

  a. Select the Microcontroller – 89V51RD2

  b. Baud Rate – 9600

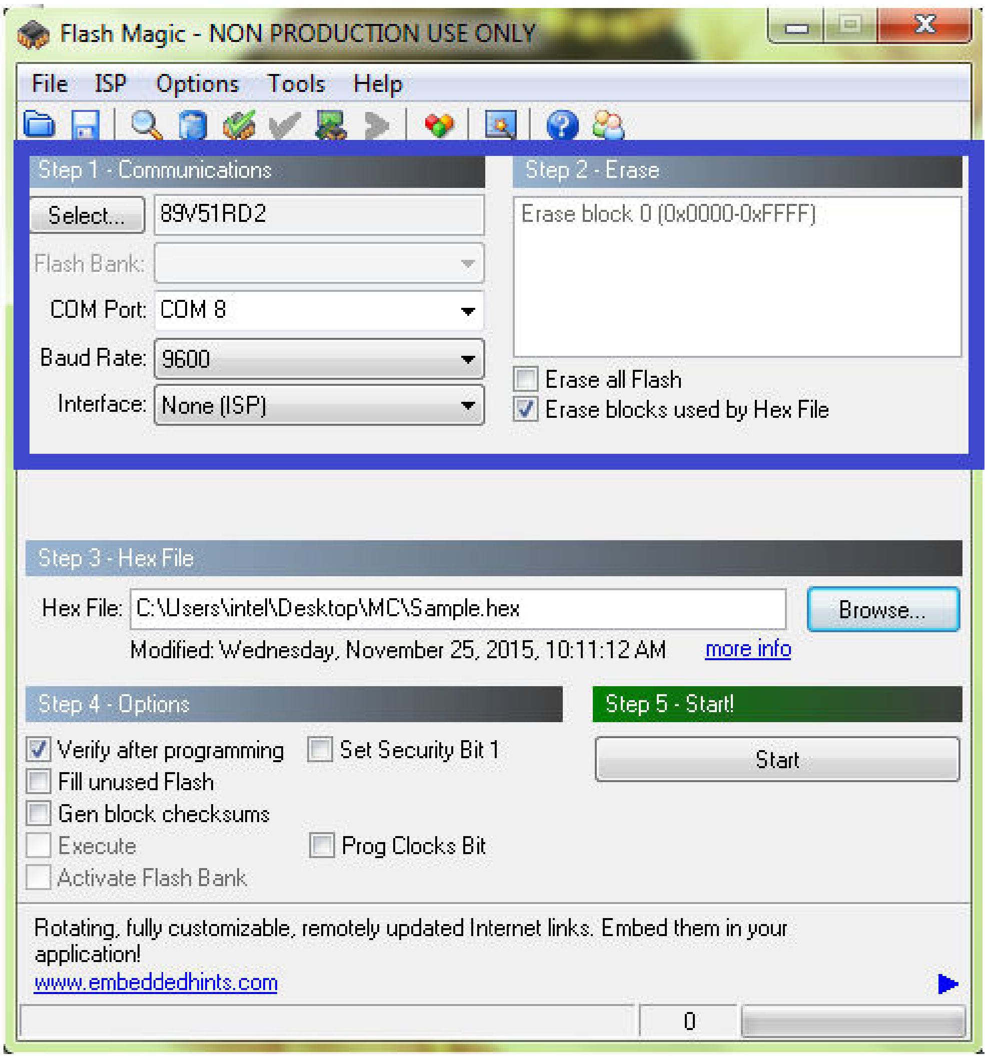
  c. COM Port – COM8

   i. Right click on My Computer ICON ,
   Select Manage → Device Manager → PORTS →Silicon Labs (COM PORT NUMBER)



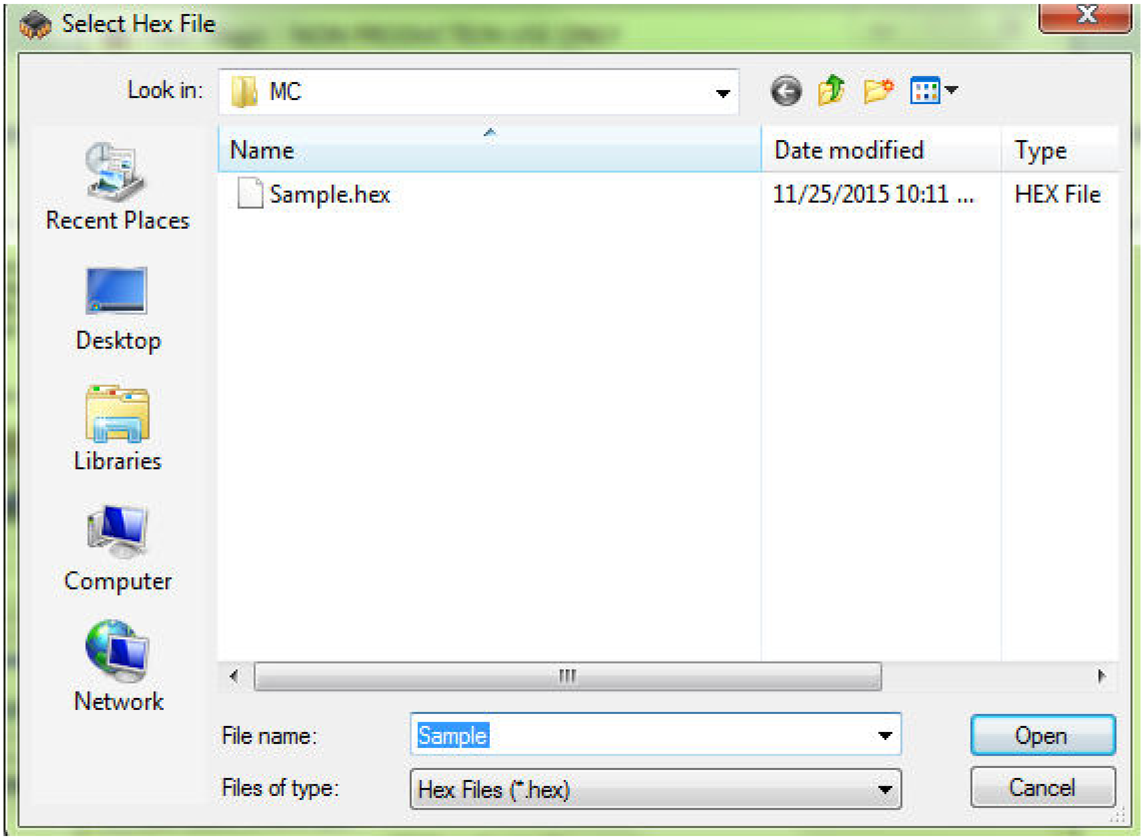
20. **In Step 2**

    a. **Tick Erase blocks used by HEX file**



21. In Step 3

a. **Click on Browse button, browse the hex file from the folder MC in the Desktop,**
b. **Select the HEX fie i.e, sample.hex and click on Open button.**

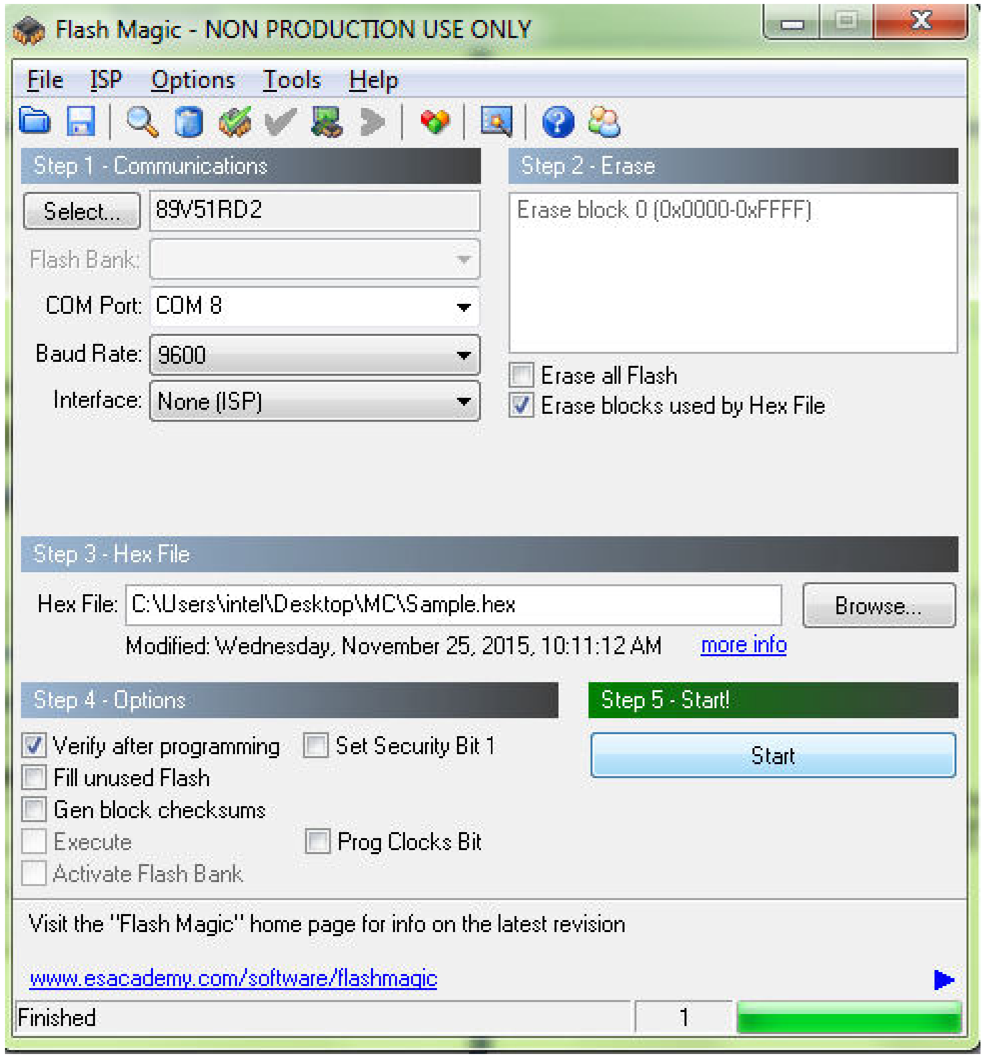


22. In Step 4

a. **Tick the option Verify after programming.**

23. In Step 5

a. **Click on Start Button which loads the HEX file on to the Microcontroller connected.**
b. **Finished appears on bottom of the Flash magic Window once done.**



24. Check the Output on the RV ALL IN ONE BOARD connected.

ASCII Table

The **American Standard Code for Information Interchange – ASCII** a character-encoding scheme originally based on the English alphabet that encodes 128 specified characters - the numbers 0-9, the letters a-z and A-Z, some basic punctuation symbols, some control codes that originated with Teletype machines, and a blank space - into the 7-bit binary integers. ASCII codes represent text in computers, communications equipment, and other devices that use text. Most modern character-encoding schemes are based on ASCII, though they support many additional characters. ASCII includes definitions for 128 characters: 33 are non-printing control characters that affect how text and space are processed and 95 printable characters, including the space.

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
0	0	NUL	16	10	DLE	32	20	(space)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	

INSTRUCTION SET

Mnemonic	Description	Byte	Cycle
Data Transfer			
MOV A,Rn	Move register to accumulator	1	1
MOV A,direct *)	Move direct byte to accumulator	2	1
MOV A,@Ri	Move indirect RAM to accumulator	1	1
MOV A,#data	Move immediate data to accumulator	2	1
MOV Rn,A	Move accumulator to register	1	1
MOV Rn,direct	Move direct byte to register	2	2
MOV Rn,#data	Move immediate data to register	2	1
MOV direct,A	Move accumulator to direct byte	2	1
MOV direct,Rn	Move register to direct byte	2	2
MOV direct,direct	Move direct byte to direct byte	3	2
MOV direct,@Ri	Move indirect RAM to direct byte	2	2
MOV direct,#data	Move immediate data to direct byte	3	2
MOV @Ri,A	Move accumulator to indirect RAM	1	1
MOV @Ri,direct	Move direct byte to indirect RAM	2	2
MOV @Ri, #data	Move immediate data to indirect RAM	2	1
MOV DPTR, #data16	Load data pointer with a 16-bit constant	3	2
MOVC A,@A + DPTR	Move code byte relative to DPTR to accumulator	1	2
MOVC A,@A + PC	Move code byte relative to PC to accumulator	1	2
MOVX A,@Ri	Move external RAM (8-bit addr.) to A	1	2
MOVX A,@DPTR	Move external RAM (16-bit addr.) to A	1	2
MOVX @Ri,A	Move A to external RAM (8-bit addr.)	1	2
MOVX @DPTR,A	Move A to external RAM (16-bit addr.)	1	2
PUSH direct	Push direct byte onto stack	2	2
POP direct	Pop direct byte from stack	2	2
XCH A,Rn	Exchange register with accumulator	1	1
XCH A,direct	Exchange direct byte with accumulator	2	1
XCH A,@Ri	Exchange indirect RAM with accumulator	1	1
XCHD A,@Ri	Exchange low-order nibble indir. RAM with A	1	1

*) MOV A,ACC is not a valid instruction

Arithmetic Operations

Mnemonic		Description	Byte	Cyc
ADD	A,Rn	Add register to Accumulator	1	1
ADD	A,direct	Add direct byte to Accumulator	2	1
ADD	A,@Ri	Add indirect RAM to Accumulator	1	1
ADD	A,#data	Add immediate data to Accumulator	2	1
ADDC	A,Rn	Add register to Accumulator with Carry	1	1
ADDC	A,direct	Add direct byte to A with carry flag	2	1
ADDC	A,@Ri	Add indirect RAM to A with Carry flag	1	1
ADDC	A,#data	Add immediate data to A with Carry flag	2	1
SUBB	A,Rn	Subtract register from A with Borrow	1	1
SUBB	A,direct	Subtract direct byte from A with Borrow	2	1
SUBB	A,@Ri	Subtract indirect RAM from A with borrow	1	1
SUBB	A,#data	Subtract immed data from A with Borrow	2	1
INC	A	Increment Accumulator	1	1
INC	Rn	Increment register	1	1
INC	direct	Increment direct byte	2	1
INC	@Ri	Increment indirect RAM	1	1
DEC	A	Decrement Accumulator	1	1
DEC	Rn	Decrement register	1	1
DEC	direct	Decrement direct byte	2	1
DEC	@Ri	Decrement indirect RAM	1	1
INC	DPTR	Increment data Pointer	1	2
MUL	AB	Multiply A and B	1	4
DIV	AB	Divide A by B	1	4
DA	A	Decimal Adjust Accumulator	1	1

Mnemonic		Description	Byte	Cycle
Program and Machine Control				
ACALL	addr11	Absolute subroutine call	2	2
LCALL	addr16	Long subroutine call	3	2
RET		Return from subroutine	1	2
RETI		Return from interrupt	1	2
AJMP	addr11	Absolute jump	2	2
LJMP	addr16	Long iump	3	2
SJMP	rel	Short jump (relative addr.)	2	2
JMP	@A + DPTR	Jump indirect relative to the DPTR	1	2
JZ	rel	Jump if accumulator is zero	2	2
JNZ	rel	Jump if accumulator is not zero	2	2
JC	rel	Jump if carry flag is set	2	2
JNC	rel	Jump if carry flag is not set	2	2
JB	bit,rel	Jump if direct bit is set	3	2
JNB	bit,rel	Jump if direct bit is not set	3	2
JBC	bit,rel	Jump if direct bit is set and clear bit	3	2
CJNE	A,direct,rel	Compare direct byte to A and jump if not equal	3	2
CJNE	A,#data,rel	Compare immediate to A and jump if not equal	3	2
CJNE	Rn,#data rel	Compare immed. to reg. and jump if not equal	3	2
CJNE	@Ri,#data,rel	Compare immed. to ind. and jump if not equal	3	2
DJNZ	Rn,rel	Decrement register and jump if not zero	2	2
DJNZ	direct,rel	Decrement direct byte and jump if not zero	3	2
NOP		No operation	1	1

Logic Operations

ANL	A,Rn	AND register to accumulator	1	1
ANL	A,direct	AND direct byte to accumulator	2	1
ANL	A,@Ri	AND indirect RAM to accumulator	1	1
ANL	A,#data	AND immediate data to accumulator	2	1
ANL	direct,A	AND accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,Rn	OR register to accumulator	1	1
ORL	A,direct	OR direct byte to accumulator	2	1
ORL	A,@Ri	OR indirect RAM to accumulator	1	1
ORL	A,#data	OR immediate data to accumulator	2	1
ORL	direct,A	OR accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive OR register to accumulator	1	1
XRL	A direct	Exclusive OR direct byte to accumulator	2	1
XRL	A,@Ri	Exclusive OR indirect RAM to accumulator	1	1
XRL	A,#data	Exclusive OR immediate data to accumulator	2	1
XRL	direct,A	Exclusive OR accumulator to direct byte	2	1
XRL	direct,#data	Exclusive OR immediate data to direct byte	3	2
CLR	A	Clear accumulator	1	1
CPL	A	Complement accumulator	1	1
RL	A	Rotate accumulator left	1	1
RLC	A	Rotate accumulator left through carry	1	1
RR	A	Rotate accumulator right	1	1
RRC	A	Rotate accumulator right through carry	1	1
SWAP	A	Swap nibbles within the accumulator	1	1

VIVA QUESTIONS

1. What is microcontroller?
2. What is the difference between microcontroller and microprocessor?
3. What is the difference between microcontroller and microcomputer?
4. Compare 8051 and MSP430 microcontrollers
5. Explain the role of watchdog timer in MSP430
6. Explain the clock system of MSP430
7. Explain the application areas of low power embedded systems.
8. List the peripherals of the microcontroller 89S8252.
9. List the special function registers of 8051 and their function
10. What is the difference between bit and byte addressable SFRs?
11. List the applications of microcontrollers.
12. What are arithmetic instructions?
13. What are logical instructions?
14. What are interrupts? List the interrupts of 8051 with their priority.
15. What do you mean by baud rate? How do to set the baud rate?
16. What is the difference between MOV and MOVC instruction?
17. What is the difference between CALL and JUMP instruction?
18. What are assembler directives?
19. What is a cross compiler?
20. What are the features of embedded C?
21. What is stack? List the instructions related to stack
22. What do you mean by data memory and code memory?
23. List out the types of memories. What do you mean by flash memory?
24. What are internal and external memories? Which instruction is used to access external memory.
25. What is the difference between timer and counter?