



JSPM UNIVERSITY, PUNE

FACULTY OF SCIENCE AND TECHNOLOGY

SCHOOL OF COMPUTATIONAL SCIENCES

Introduction to Embedded Systems Lab Manual (230GETB41)

B.Tech in _____

Name: _____

PRN: _____ Roll No: _____

Branch: _____ Division: _____

2025 - 2026



JSPM UNIVERSITY PUNE

Recognized by UGC u/s 2 (f) of UGC Act 1956 and enacted by the
State Government of Maharashtra - JSPM University Act, 2022 (Mah.IV of 2023)

JSPM University Pune B.Tech. "CSE" V/VII Sem

Course Type: MMC	Lab Course Title: Introduction to Embedded System Lab	
Course Code: 230GETB41	Teaching Scheme: (Hrs./Week)	Examination Scheme:
Credits: 1	Lecture (L): Tutorial (T): Practical (P): 2 Experiential Learning (EL):	Theory (TH): -- Practical (PR): 50 Marks Oral (OR): 50 Marks

Prerequisite Courses, if any: -

List of Laboratory Experiments

[Conduct the following experiments by writing program using 8051/AT89C51 using an evaluation board/simulator and the required software tool]

- 1 Write a program to find the GCD and LCM of a given two bytes in a memory locations
- 2 Write a program to convert the given BCD number into its equivalent seven segment value.
- 3 Toggling of p1.5 with the delay of 5ms without interrupts using timer 1
- 4 Toggling of p1.2 with the delay of 5ms with interrupts using timer 0
- 5 Count no. Of pulses without interrupts using counter 0
- 6 Count no. Of pulses with interrupts using counter 1
- 7 Program to transfer data serially with baud rate of 9600
- 8 Program to receive data serially with baud rate of 9600
- 9 Sequence Generator Using Serial Interface in 8051
- 10 Programming for Delay Generation and Effect of CPU Clock
- 11 Programming with On-Chip Timers/Counters
- 12 Programming with On-Chip ADC
- 13 Programming for PWM Generation
- 14 Programming for I/O Interfacing (LED and Switch Interfacing)

Virtual LAB Links:

1. Lab Name: Embedded Systems Link of the Virtual Lab:<https://esd-coep.vlabs.ac.in/introduction.html>
2. Lab Name: Real Time Embedded Systems Laboratory
Link of the Virtual Lab: <http://vlabs.iitkgp.ac.in/rtes/#>

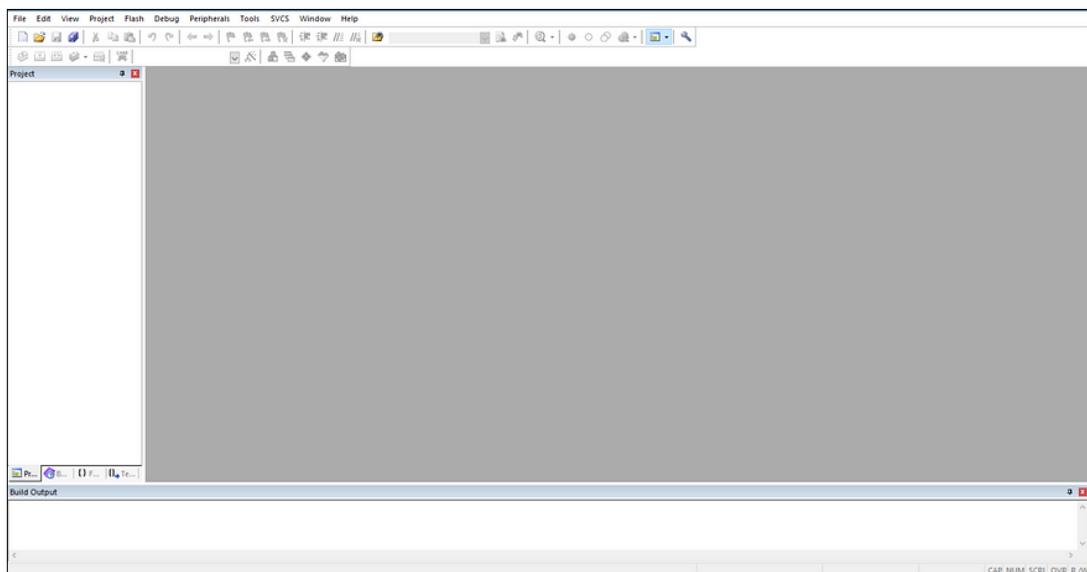


INTRODUCTION TO KEIL µvision

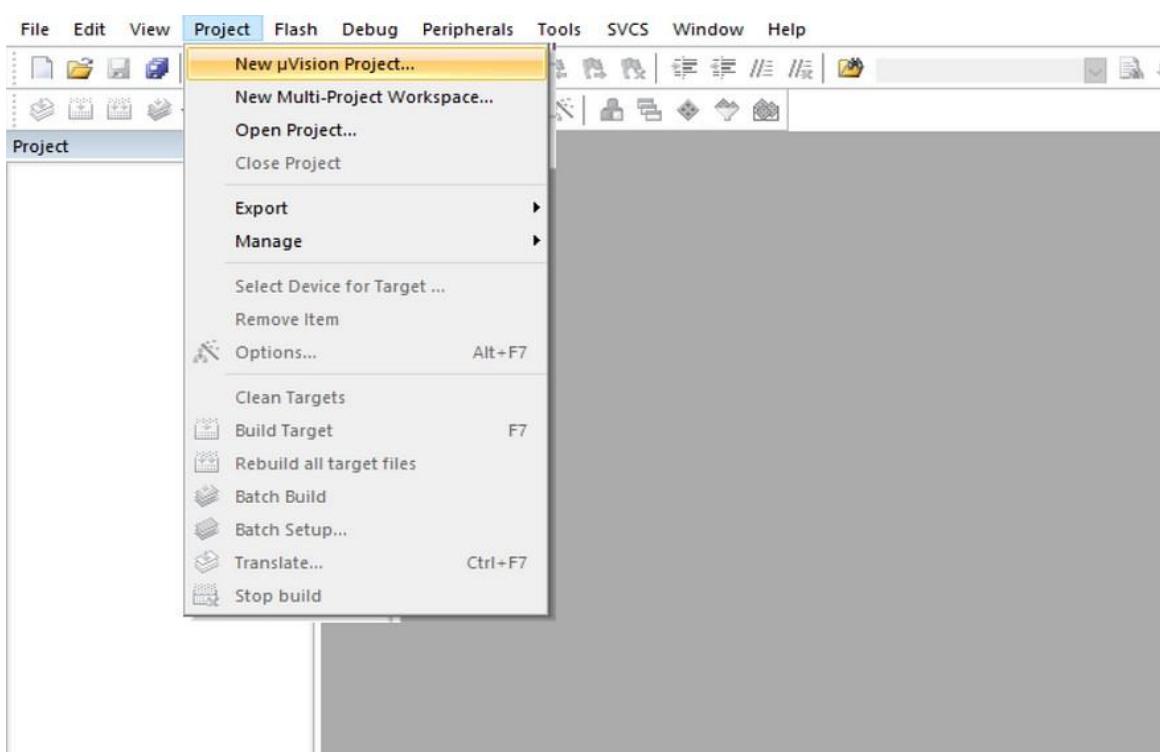
We will explore how to program the 8051 microcontroller using Keil uVision5.

Creating a Project in Keil µvision

When we open Keil µvision for first time we will be able to see IDE as shown in figure below.

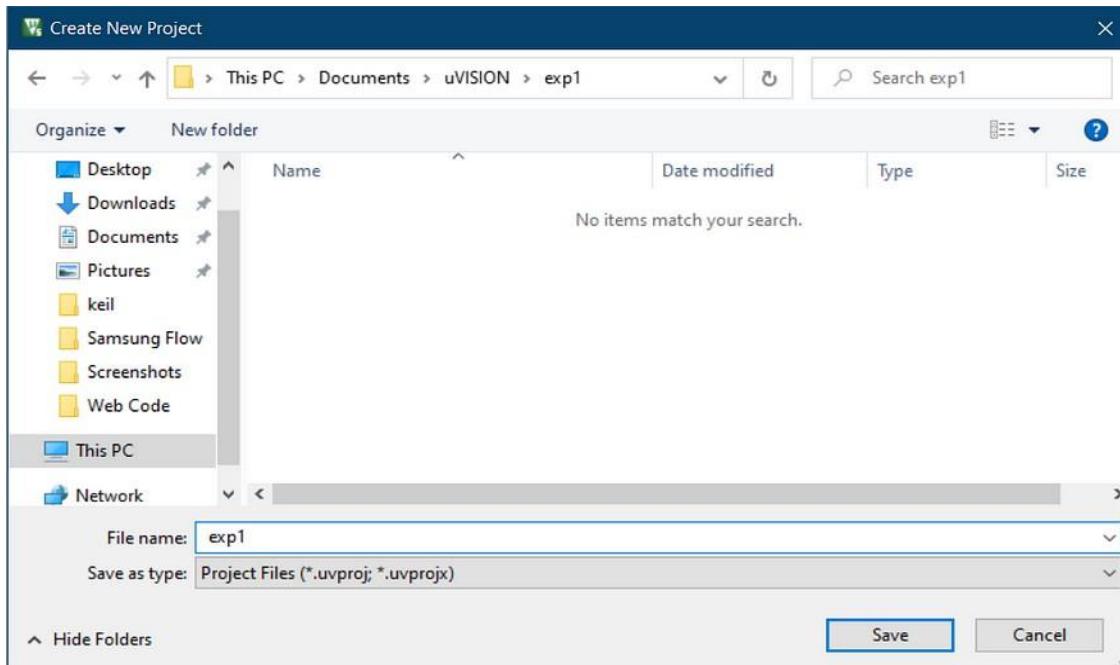


To create a new 8051 project using Keil IDE, click on the 'Project' item on the IDE Menu bar and select 'New uVision Project...' as shown in the above image.



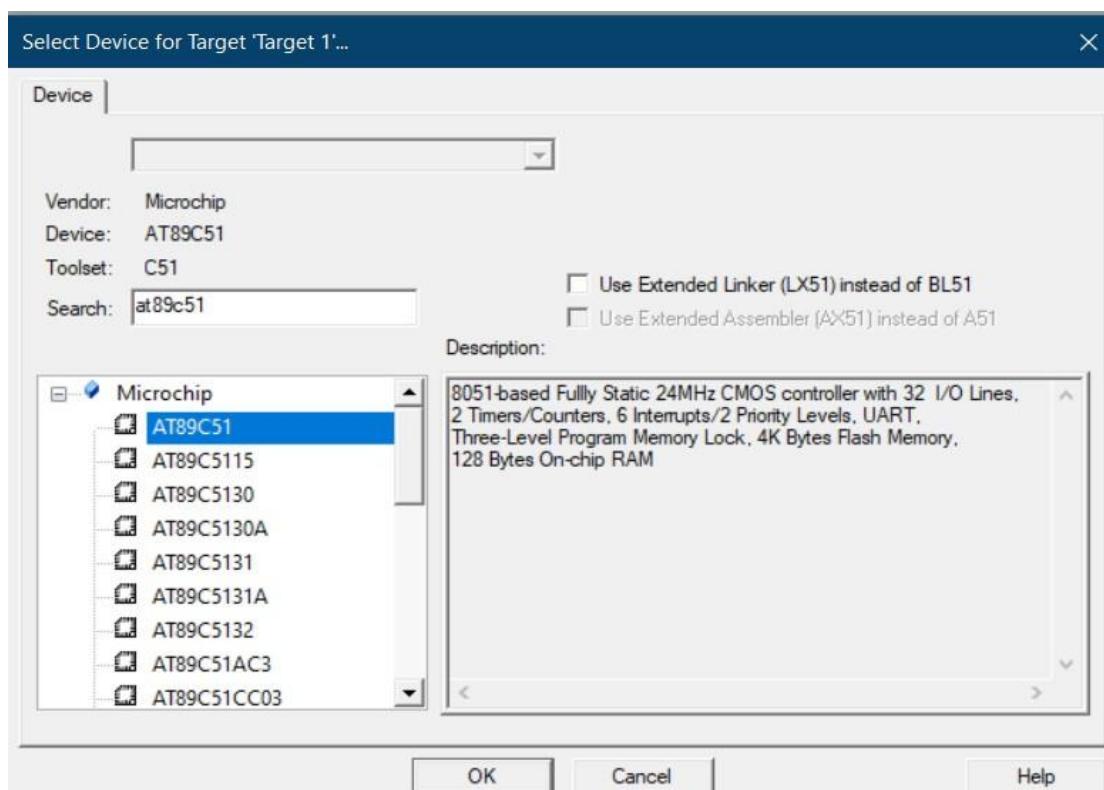


Now create a Folder to store your project and give a name to your Project files (*.uvproj), for eg exp1 (exp1.uvproj) , then save it.



Selecting 8051 Microcontroller

- After saving the project , a dialog box will appear to select a Device
- Then type “AT89C51” in search box come under Microchip
- Click on AT89C51 and click on OK

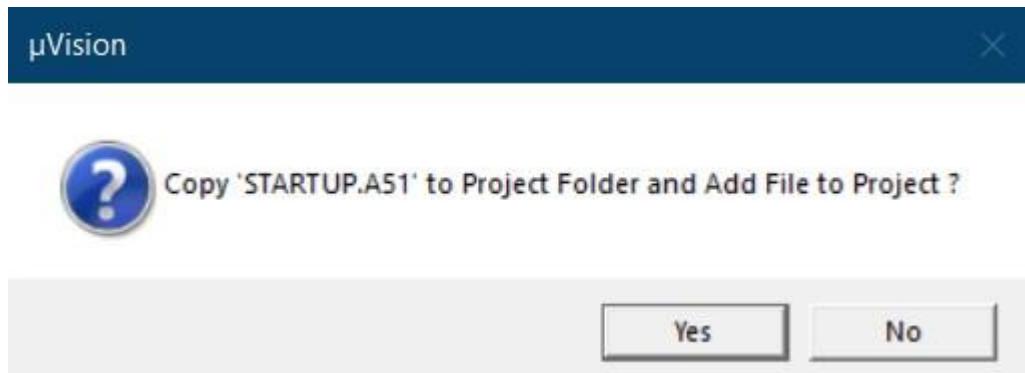




JSPM UNIVERSITY PUNE

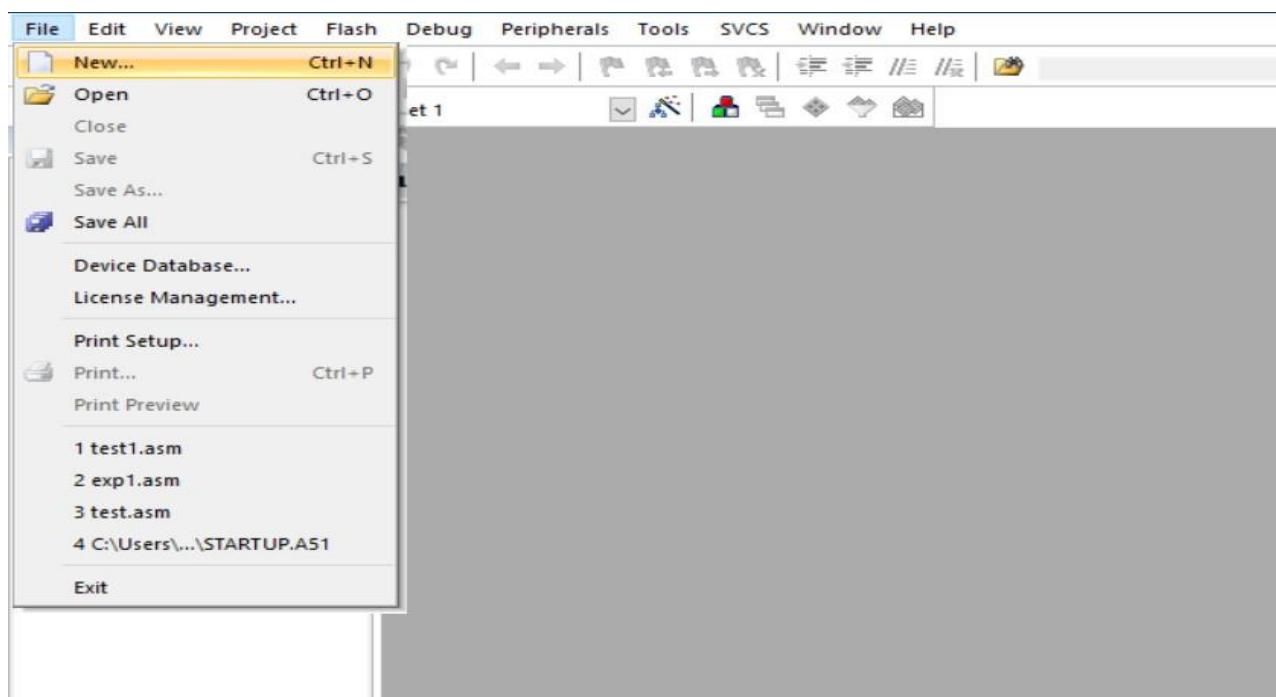
Recognized by UGC u/s 2 (f) of UGC Act 1956 and enacted by the
State Government of Maharashtra - JSPM University Act, 2022 (Mah.IV of 2023)

You will get another dialog as shown below. Asking to copy STARTUP.A51, Click on No



Writing Code

Now in order to write code we need a blank file, Click on File >> New... or can use keyboard shortcut Ctrl+ N





JSPM UNIVERSITY PUNE

Recognized by UGC u/s 2 (f) of UGC Act 1956 and enacted by the State Government of Maharashtra - JSPM University Act, 2022 (Mah.IV of 2023)

Write a code as shown in the figure below

```
$mod51
org 00h
mov dptr,#4300h
mov r2,#0ah
mov r1,#50h
mov r0,#00h
loop:mov dpl,r0
    movx a,@dptr
    inc r0
    mov dpl,r1
    movx @dptr,a
    inc r1
    djnz r2,loop
l1:sjmp l1
end
```

Now “Save As” the file , Click on File >> Save As...

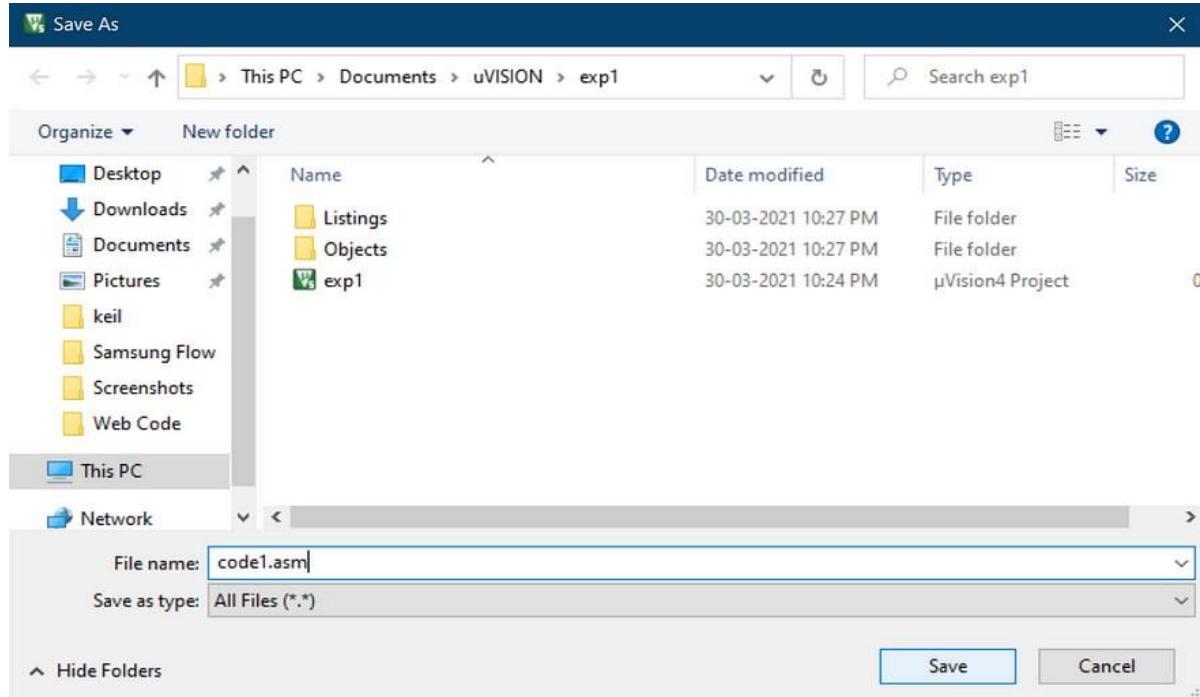
```
$mod51
org 00h
mov dptr,#4300h
mov r2,#0ah
mov r1,#50h
mov r0,#00h
loop:mov dpl,r0
    movx a,@dptr
    inc r0
    mov dpl,r1
    movx @dptr,a
    inc r1
    djnz r2,loop
l1:sjmp l1
end
```



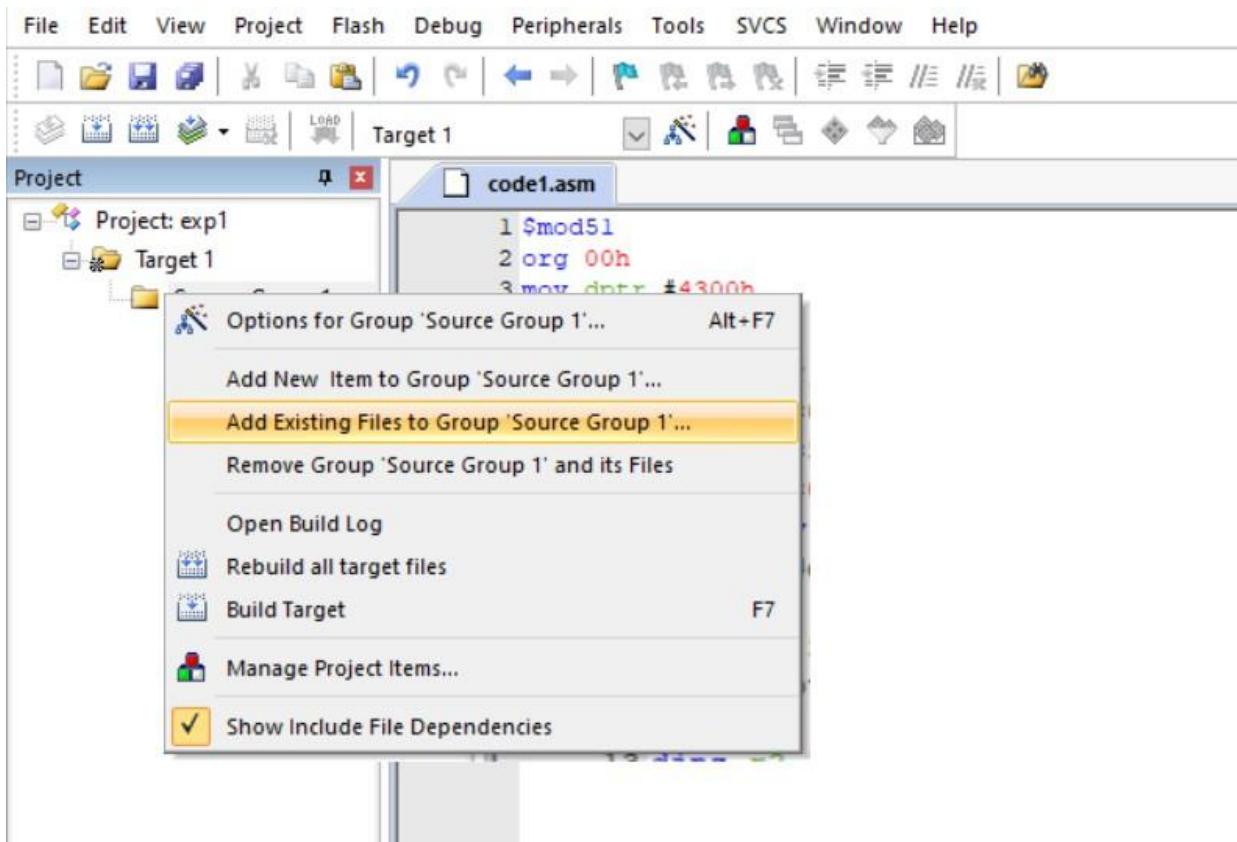
JSPM UNIVERSITY PUNE

Recognized by UGC u/s 2 (f) of UGC Act 1956 and enacted by the State Government of Maharashtra - JSPM University Act, 2022 (Mah.IV of 2023)

Give a file name to your code with extension *.asm, for eg code.asm , then save it.



In left corner you can see a Project Box , in that Project Box expand Target1 , then Right click on Source Group 1, Then click on Add Existing Files to Group 'Source Group 1'...

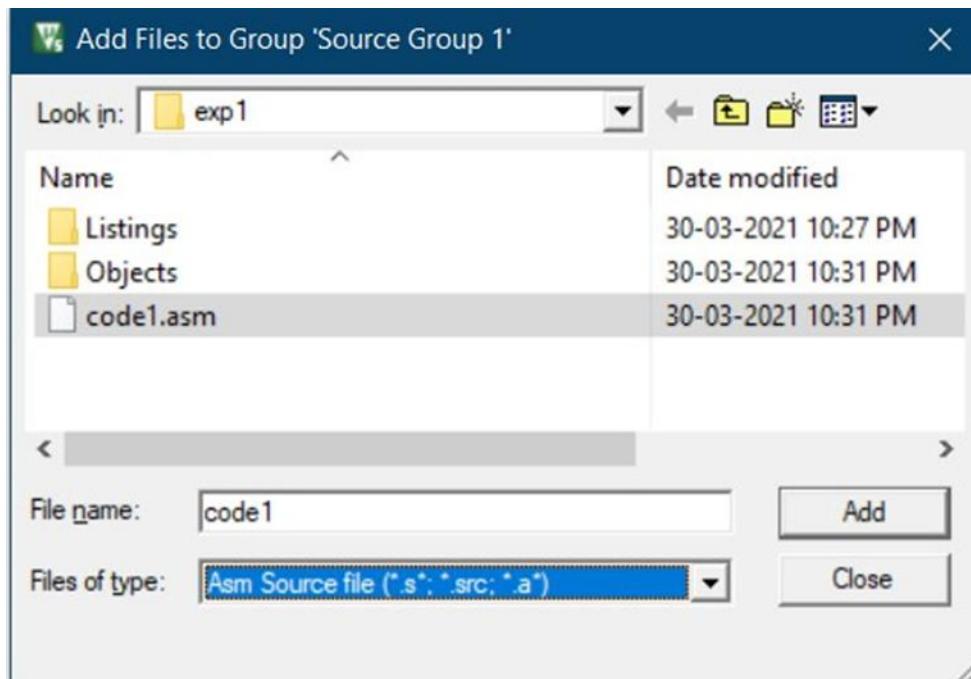




JSPM UNIVERSITY PUNE

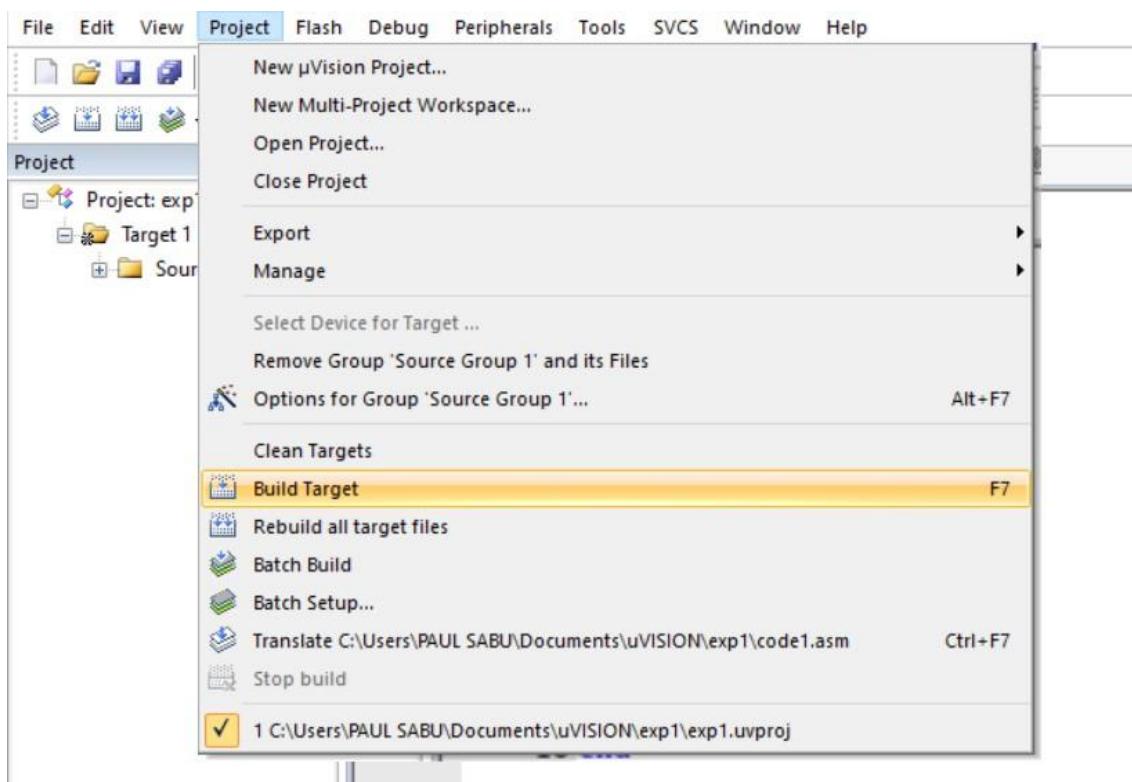
Recognized by UGC u/s 2 (f) of UGC Act 1956 and enacted by the State Government of Maharashtra - JSPM University Act, 2022 (Mah.IV of 2023)

Select the Files of type : Asm Source file (*.s*; *.src*; *.a*) , Select the file with extenstion *.asm , here in my case it is "code1.asm" , Click on Add and click on Close



Project Build

To Bulid Project , Click on Project >> Build Target or press on F7 key



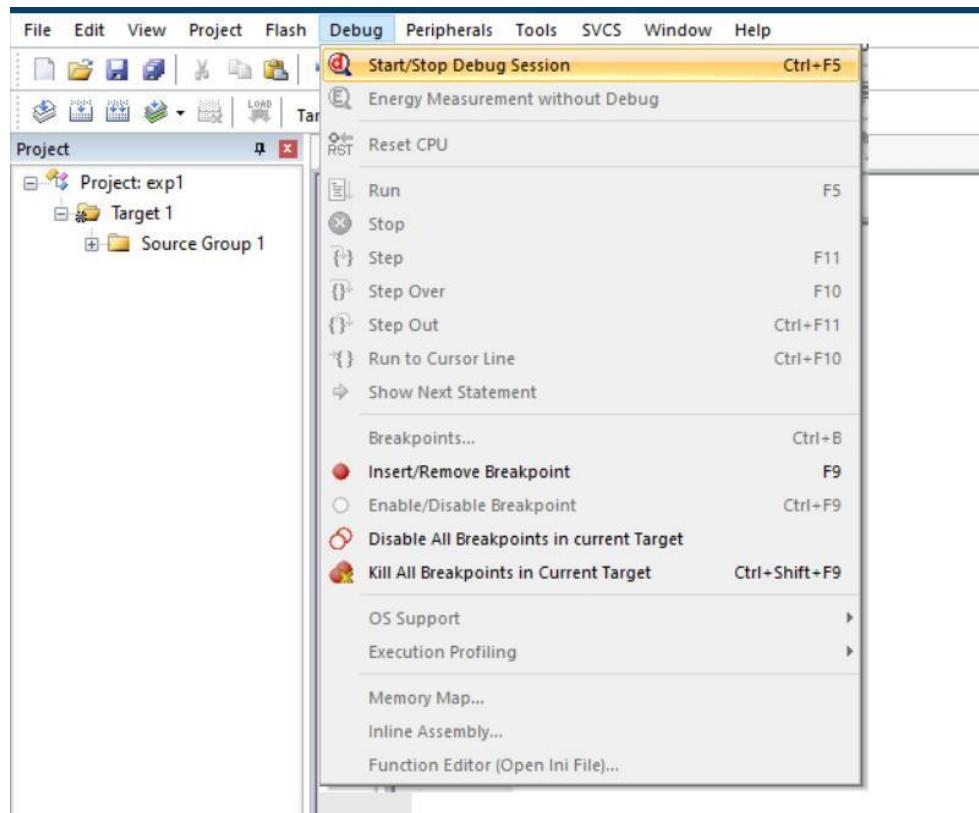


If the Build is successful there will not be any error or any warning as shown in the figure below , else verify the code and rebuild it.

```
Build Output
linking...
Program Size: data=8.0 xdata=0 code=21
".\Objects\expl" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:00
```

Debug and Run Code

Next step is to debug the code, For that click on Debug >> Start/StopDebugSession or press on Ctrl+ F5

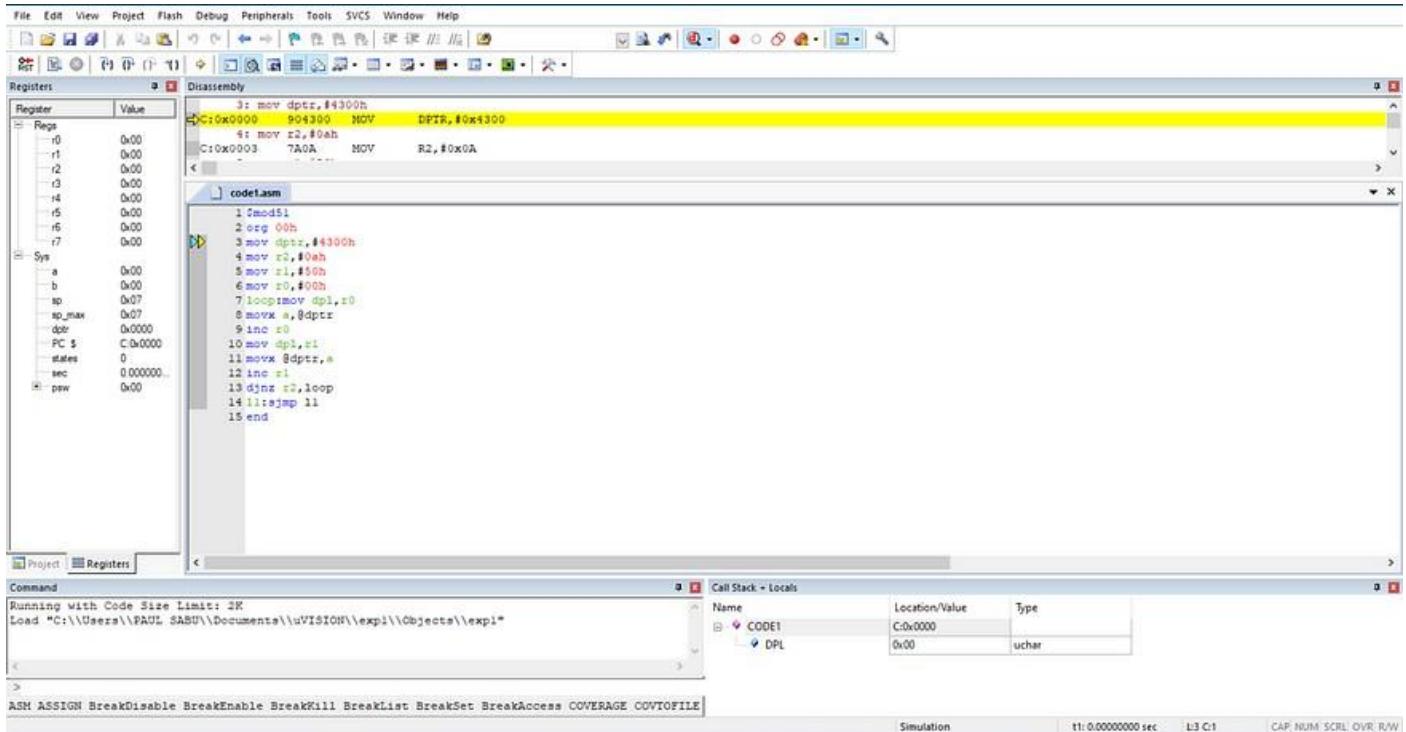




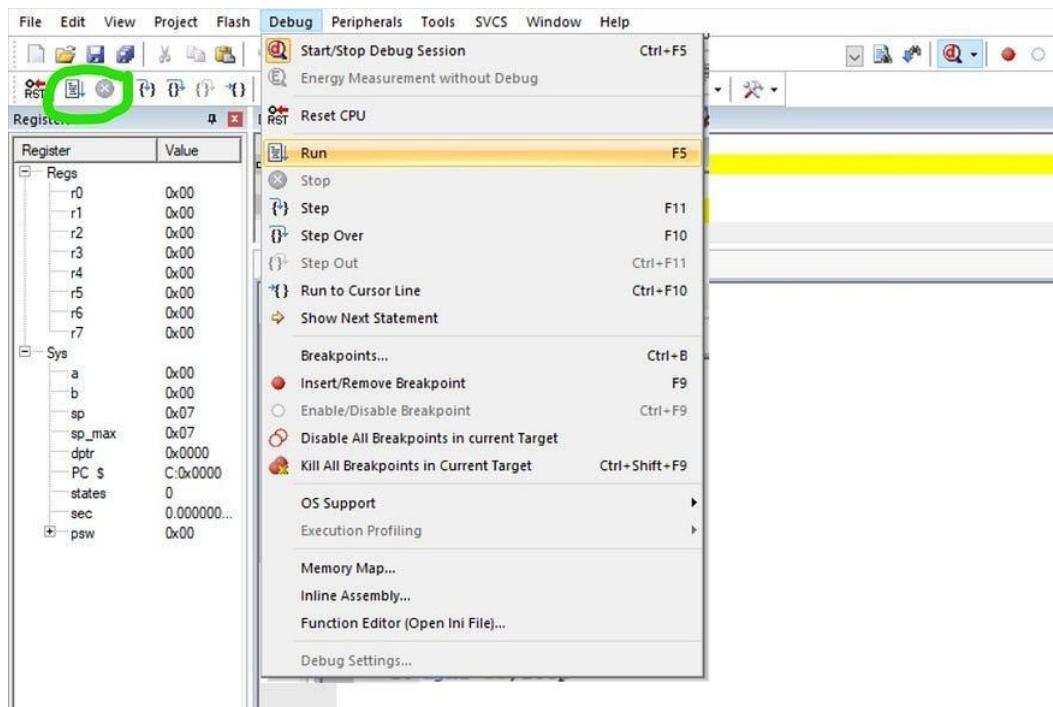
JSPM UNIVERSITY PUNE

Recognized by UGC u/s 2 (f) of UGC Act 1956 and enacted by the State Government of Maharashtra - JSPM University Act, 2022 (Mah.IV of 2023)

After Entering to debug , you will have a similar screen as shown in the figure below



Next step is to Run the code , For that click on Debug >> Run or press on F5 or click "Run" button highlighted in green color

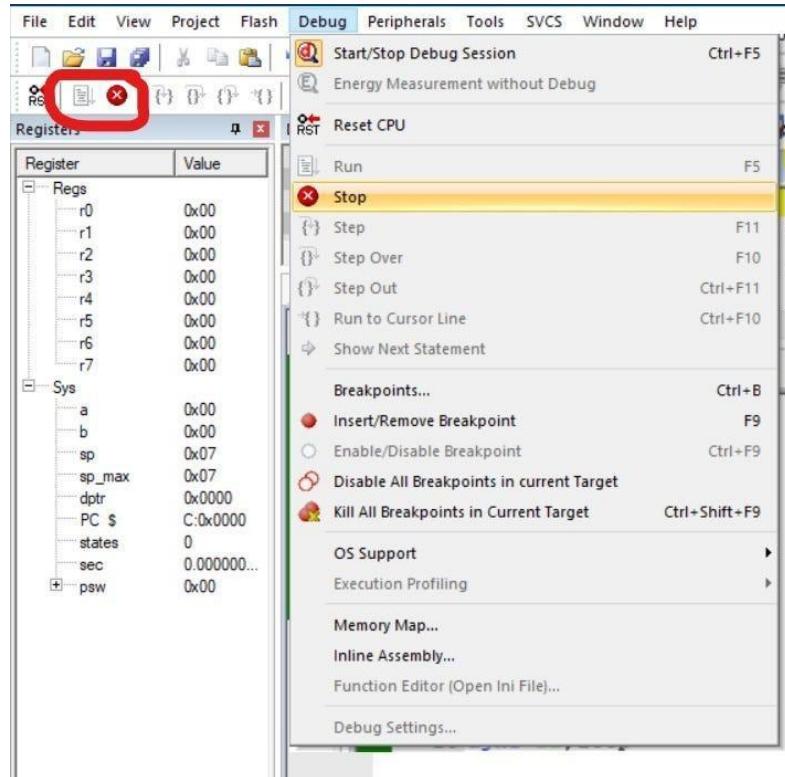


After a few seconds we need to Stop running the code , For that click on Debug >> Stop or click "Stop" button highlighted in red color

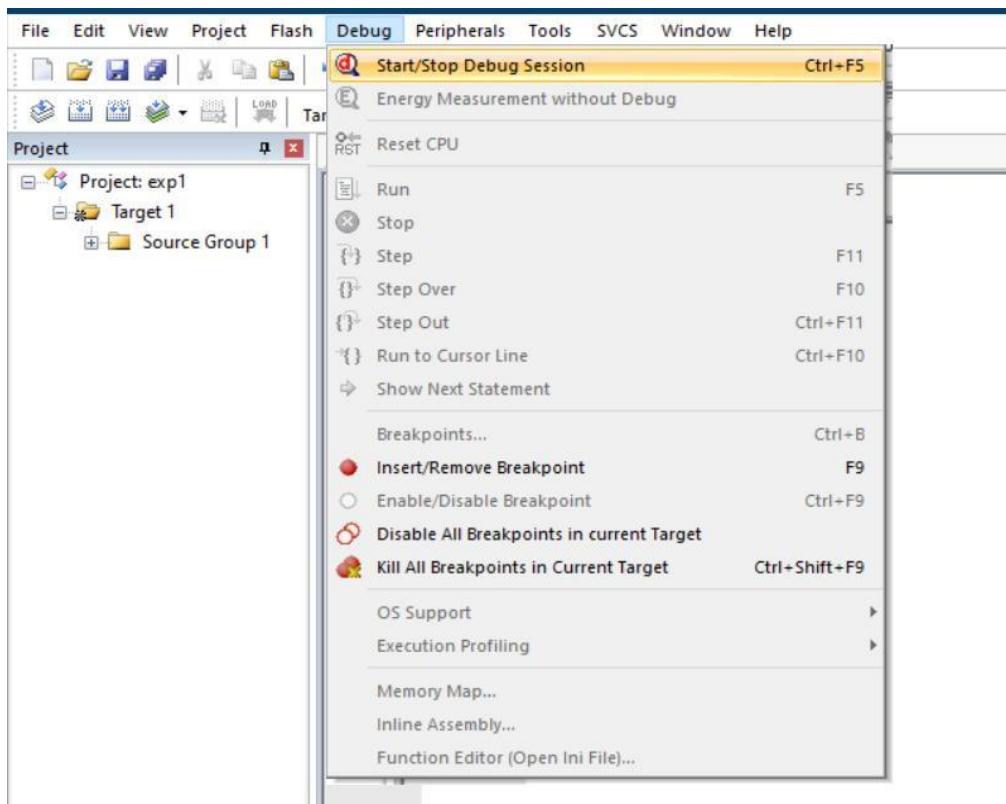


JSPM UNIVERSITY PUNE

Recognized by UGC u/s 2 (f) of UGC Act 1956 and enacted by the State Government of Maharashtra - JSPM University Act, 2022 (Mah.IV of 2023)



Before Closing the Keil uVision We need to stop Debugging, for that click on Debug >> Start/Stop Debug Session or press on Ctrl+ F5, Then close it





Experiment - 1

AIM:

- Write an assembly language program to find the GCD and LCM of a given two bytes in a memory locations.

SOFTWARE REQUIRED: Keil μvision

ALGORITHM:

Algorithm for GCD (Greatest Common Divisor)

Let the two input numbers be A and B.

Step 1: Start

Step 2: Read two positive integers A and B

Step 3: Repeat the following until B becomes 0:

- Find the remainder $R = A \bmod B$
- Assign $A = B$
- Assign $B = R$

Step 4: When $B = 0$, the value of A is the GCD

Step 5: Print GCD

Step 6: Stop

Example (GCD of 48 and 18)

A	B	A mod B (R)
48	18	12
18	12	6
12	6	0

$$\text{GCD} = 6$$

Algorithm for LCM (Least Common Multiple)



Step 1: Start

Step 2: Read two positive integers A and B

Step 3: Find GCD of A and B using the GCD algorithm above

Step 4: Compute LCM = (A × B) / GCD

Step 5: Print LCM

Step 6: Stop

Example (LCM of 12 and 18)

$$\text{GCD}(12, 18) = 6$$

$$\text{LCM} = (12 \times 18) / 6 = 216 / 6 = \mathbf{36}$$

CODE:

Program	Comments
a) Program to find GCD and LCM ORG 0000H MOV A, 50H MOV B, 51H BACK: MOV R1,B DIV AB MOV A, B CJNE A, #00H, L1 L1: JZ LAST MOV A, R1 SJMP BACK LAST: MOV A, R1 MOV 52H, A MOV A, 50H MOV B, 51H MUL AB MOV B, 52H DIV AB MOV 53H, A END	Starting address of Memory (Origin) First number is in 50H moved to register A Second number is in 51H moved to register B Second number moved from B to R1 Register Divide A/B (Result The quotient is stored in register A, and the remainder is stored in register B after the division), Remainder is compared with zero If zero ,go to last and terminate; otherwise repeat Store the GCD value in 52H First number is in 50H moved to register A Second number is in 51H moved to register B Multiply A and B result stored in A Move GCD to B register Divide A/B Quotient will be LCM and store in 53H End of the program

RESULT:

Before Execution:



JSPM UNIVERSITY PUNE

Recognized by UGC u/s 2 (f) of UGC Act 1956 and enacted by the
State Government of Maharashtra - JSPM University Act, 2022 (Mah.IV of 2023)

```
Memory 1
Address: d:50h
D:0x50: 04 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D:0x61: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Call Stack + Locals | Memory 1
```

After Execution:

```
Memory 1
Address: d:50h
D:0x50: 04 06 02 0C 00 00 00 00 00 00 00 00 00 00 00 00
D:0x61: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Conclusion: We have written ALP program for GCD and LCM of two bytes and verified the output using Keil μvision Software



Experiment - 2

AIM:

Write a program to convert the given BCD number into its equivalent seven segment value.

SOFTWARE REQUIRED: Keil μvision

ALGORITHM:

Algorithm for BCD equivalent seven segment value

Step 1: Start

Step 2: Initialize Port 1 (P1) as an input port and Port 2 (P2) as an output port, potentially to control a 7-segment display.

Step 3: Specifies the starting address of the lookup table in program memory by Initializing the Data Pointer.

Step 4: Main Loop (AGAIN): Read Input from Port 1 (P1) to the Accumulator (A) and Mask Lower Nibble by Performing a bitwise AND operation

Step 5: Use the value in the Accumulator as an offset to access a value from the lookup table pointed to by DPTR.

Step 6: Move the 7-segment pattern from the Accumulator (A) to Port 2 (P2) to illuminate the corresponding segments of the 7-segment display.

Step 7: Continuously read and display the input.

Step 8: Stop

Example (BCD equivalent seven segment value)

Lookup table: Seven-segment codes

3FH - Represents the digit 0.

06H - Represents the digit 1.



5BH - Represents the digit 2.

4FH - Represents the digit 3.

66H - Represents the digit 4.

6DH - Represents the digit 5.

7DH - Represents the digit 6.

07H - Represents the digit 7.

7FH - Represents the digit 8.

6FH - Represents the digit 9.

CODE:

Program	Comments
BCD equivalent seven segment value	
ORG 0000H	Starting address of Memory (Origin)
MOV P1, #0FFH	Port 1 as an Input
MOV P2, #00H	Port 2 as an output
MOV DPTR, #0300H	Initialize the DPTR for the segment table
AGAIN: MOV A, P1	move the input value to the accumulator
ANL A, #0FH	Mask the lower nibble of the accumulator
MOVC A,@A+DPTR	Accumulator as an offset to access a value from the lookup table pointed to by DPTR.
MOV P2, A	move the accumulator's input value to Port 2
SJMP AGAIN	Loop to continuously read and display the input
ORG 0300H	Starting address of the lookup table stored in program memory
DB	
3FH,06H,5BH,4FH,66H,6DH,7DH,07H, 7FH,6FH	Seven-segment codes for BCD
END	End of the program

RESULT:



JSPM UNIVERSITY PUNE

Recognized by UGC u/s 2 (f) of UGC Act 1956 and enacted by the State Government of Maharashtra - JSPM University Act, 2022 (Mah.IV of 2023)

Before Execution:

The screenshot shows the µVision IDE interface. The assembly code in the Etb2.asm file is as follows:

```
        4: MOV Dptr, #0300H
C:0x0006 900300    MOV      Dptr,#0x0300
        5: AGAIN: MOV A, P1
C:0x0009 E590    MOV      A,P1(0x90)
        —
1 ORG 0000H
2 MOV P1, #0FFH
3 MOV P2, #00H
4 MOV Dptr, #0300H
5 AGAIN: MOV A, P1
6 ANL A, #0FH
7 MOVC A,@A+Dptr
8 MOV P2, A
9 SJMP AGAIN
10 ORG 0300H
11 DB 3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH
12 END
```

The Registers window shows the initial values of various registers. The Disassembly window highlights the instruction at address C:0x0009. Two configuration windows are open: Parallel Port 1 and Parallel Port 2. Parallel Port 1 is set to Port 1 (P1: 0xFF, 7 Bits, 0) and Parallel Port 2 is set to Port 2 (P2: 0x00, 7 Bits, 0).

After Execution: Input BCD Value 1 on port 1, the Seven-segment code is 6 on port 2

The screenshot shows the µVision IDE interface after execution. The assembly code in the Etb2.asm file has been modified to reflect the execution:

```
        9: SJMP AGAIN
C:0x0010 80E7    SJMP    AGAIN(C:0009)
C:0x0012 00      NOP
C:0x0013 00      NOP
        —
1 ORG 0000H
2 MOV P1, #0FFH
3 MOV P2, #00H
4 MOV Dptr, #0300H
5 AGAIN: MOV A, P1
6 ANL A, #0FH
7 MOVC A,@A+Dptr
8 MOV P2, A
9 SJMP AGAIN
10 ORG 0300H
11 DB 3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH
12 END
```

The Registers window shows updated values. The Disassembly window highlights the instruction at address C:0x0010. The configuration windows show Parallel Port 1 (Port 1: 0x01, 7 Bits, 0) and Parallel Port 2 (Port 2: 0x06, 7 Bits, 0). The pins for Parallel Port 2 are shown as 0x06.

Conclusion:



Experiment-3

AIM:

Toggling of p1.5 with the delay of 5ms without interrupts using timer 1

SOFTWARE REQUIRED: Keil µvision

ALGORITHM:

Algorithm for delay of 5ms without interrupts using timer 1

Step 1: Start

Step 2: Configure Timer 1: Load the TMOD register to set Timer 1 to operate in Mode 1 (16-bit timer mode).

Step 3: Load Timer 1 registers with the desired delay value into the Timer 1 registers, TH1 (high byte) and TL1 (low byte). Toggle P1.5 by completing the current state of pin P1.5.

Step 4: Start Timer 1 by using Set Timer 1 Run bit

Step 5: Continuously check the Timer 1 Overflow Flag (TF1) until it is set (becomes 1).

Step 6: Timer Overflow Actions: Stop Timer 1 by clearing the TR1 control bit

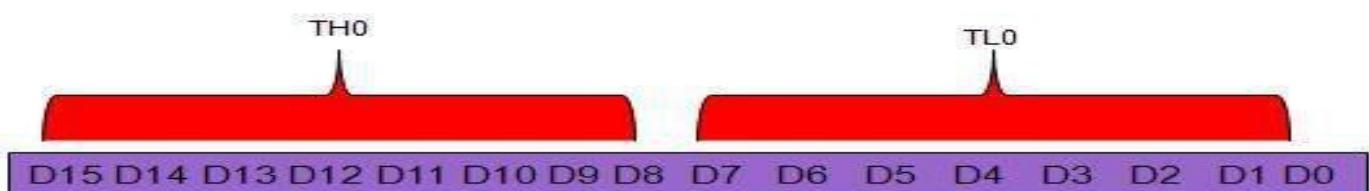
Clear Timer 1 Overflow Flag by Reset the TF1 flag to prepare for the next timer cycle

Step 7: Loop Back and Return to the beginning of the AGAIN loop to repeat the entire process

Step 8: Stop

All Timer Registers(SFR):

TIMER 0 Register:



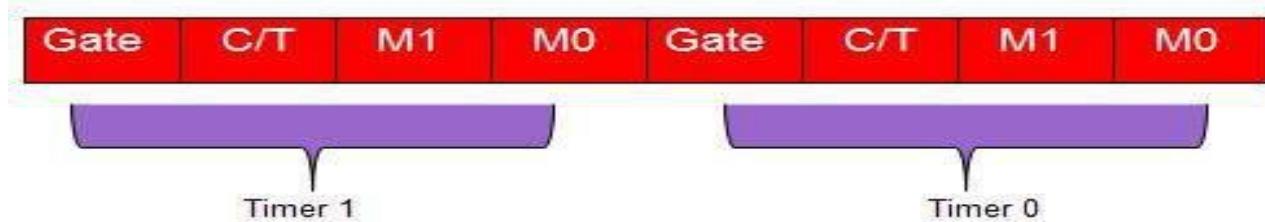
Same for the timer 1



JSPM UNIVERSITY PUNE

Recognized by UGC u/s 2 (f) of UGC Act 1956 and enacted by the State Government of Maharashtra - JSPM University Act, 2022 (Mah.IV of 2023)

TMOD (Timer Mode) Register



Gate – When set, the timer only runs while INT(0,1) is high.

C/T – Counter/Timer select bit.

For Counter 1 and for Timer 0

M1 – Mode bit 1.

M0 – Mode bit 0.

M1	M2	Mode
0	0	13-bit timer mode.
0	1	16-bit timer mode.
1	0	8-bit auto reload mode.
1	1	Spilt mode.

TCON (Timer Control Register)



Name	Description	
7	TF1 (Timer 1 Overflow Flag)	Set to 1 when Timer 1 overflows else 0.
6	TR1 (Timer 1 Run Control Bit)	Set to 1 to start Timer 1, and set to 0 to stop Timer 1.
5	TFO (Timer 0 Overflow Flag)	Set to 1 when Timer 0 overflows else 0.
4	TRO (Timer 0 Run Control Bit)	Set to 1 to start Timer 0, and set to 0 to stop Timer 0.
3	IE1 (Interrupt 1 Edge Flag)	Set to 1 when an external interrupt 1 occurs else 0.
2	IT1 (Interrupt 1 Type Control Bit)	Set to 1 to configure external interrupt 1 as edge-triggered, and set to 0 for level-triggered.
1	IE0 (Interrupt 0 Edge Flag)	Set to 1 when an external interrupt 0 occurs else 0.
0	ITO (Interrupt 0 Type Control Bit)	Set to 1 to configure external interrupt 0 as edge-triggered, and set to 0 for level-triggered.



JSPM UNIVERSITY PUNE

Recognized by UGC u/s 2 (f) of UGC Act 1956 and enacted by the State Government of Maharashtra - JSPM University Act, 2022 (Mah.IV of 2023)

CODE:

Program	Comments
<p><i>Toggling of p1.5 with the delay of 5ms without interrupts using timer 1</i></p> <pre>ORG 0000H MOV TMOD, #10h AGAIN: MOV TL1, #0FFH MOV TH1, #0EDH CPL P1.5 SETB TR1 BACK: JNB TF1, BACK CLR TR1 CLR TF1 SJMP AGAIN END</pre>	<p>Set Timer 1 to operate in Mode 1 (16-bit timer mode).</p> <p>Load Timer 1 registers with the desired delay value</p> <p>Toggle P1.5 by completing the current state</p> <p>Set Timer 1 Run bit</p> <p>Continuously check the Timer 1 Overflow</p> <p>Clearing the Timer run(Stop the timer)</p> <p>Clearing the Overflow(for the next timer cycle))</p> <p>Repeat the entire process</p>

RESULT:

Before Execution:

The screenshot shows a debugger interface with the following components:

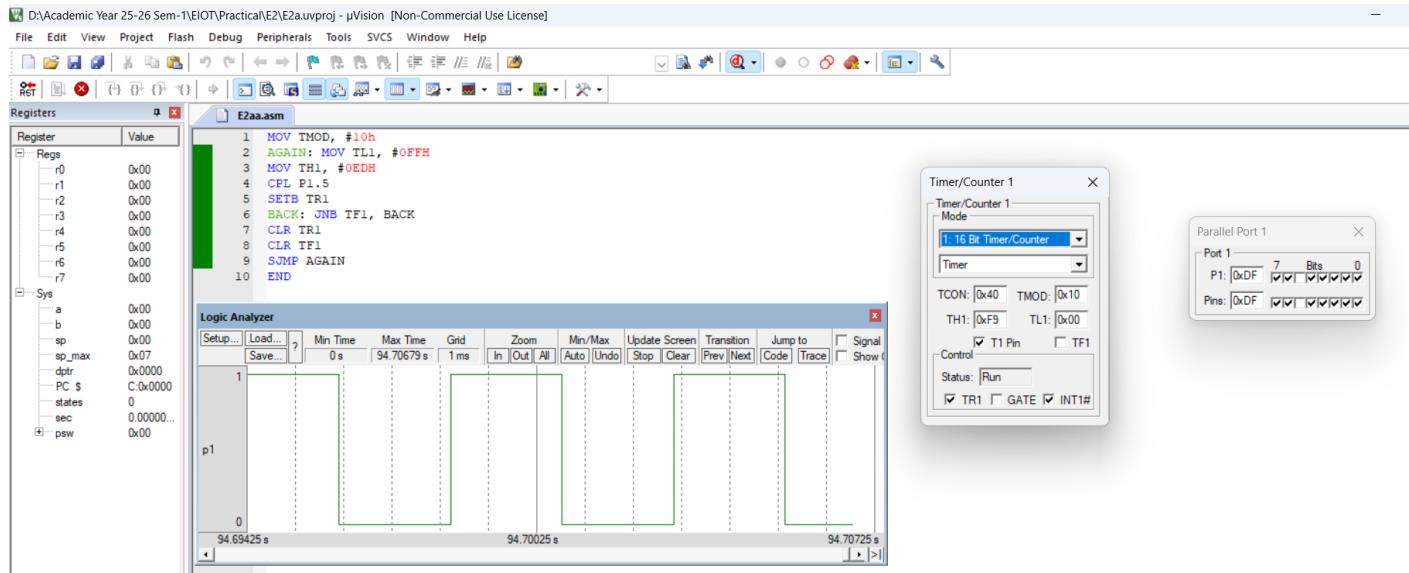
- File Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help.
- Toolbar:** Includes icons for file operations, project management, and debugging.
- Registers Window:** Shows the register state for Registers (r0-r7) and System (a, b, sp, etc.).
- Code Editor:** The assembly code for the program:1 MOV TMOD, #10h
2 AGAIN: MOV TL1, #0FFH
3 MOV TH1, #0EDH
4 CPL P1.5
5 SETB TR1
6 BACK: JNB TF1, BACK
7 CLR TR1
8 CLR TF1
9 SJMP AGAIN
10 END
- Timer/Counter 1 Dialog:** Configuration for Timer/Counter 1:
 - Mode: 0:13 Bit Timer/Counter
 - Timer
 - TCON: 0x00, TMOD: 0x00
 - TH1: 0x00, TL1: 0x00
 - T1 Pin checked, TF1 unchecked
 - Status: Stop
 - Control options: TR1, GATE, INT1#
- Parallel Port 1 Dialog:** Configuration for Port 1:
 - P1: 0xFF, 7 Bits, 0
 - Pins: 0xFF



JSPM UNIVERSITY PUNE

Recognized by UGC u/s 2 (f) of UGC Act 1956 and enacted by the
State Government of Maharashtra - JSPM University Act, 2022 (Mah.IV of 2023)

After Execution:



Conclusion:



Experiment-4

AIM:

Toggling of p1.2 with the delay of 5ms with interrupts using timer 0

SOFTWARE REQUIRED: Keil µvision

ALGORITHM:

Algorithm for delay of 5ms with interrupts using timer 0

Step 1: Start(Main Program)

Step 2: Configur Timer 0 to operate in Mode 1 (16-bit timer mode).

Step 3: Load Timer 1 registers with the desired delay value into the Timer 0 registers, TH0 (high byte) and TL0 (low byte).

Step 4: Enable the global interrupt by setting the EA bit in the IE register.

Step: 5: Enable the Timer 0 interrupt by setting the ET0 bit in the IE register.

Step 6: Start Timer 0: Set the TR0 bit in the TCON register to start Timer 0 and Enter an infinite loop

Step 7: END

Timer 0 Interrupt Service Routine (ISR)

This routine is automatically executed when Timer 0 overflows (reaches its maximum count and resets to 0) and the interrupt is enabled:

Step 1: Select the address of Interrupt T0 (000B)

Step 2: Toggle P1.2: Complement the state of the P1.2 pin

Step 3: Reload the registers TL and TH with initial count

Step 4: RETI instruction returns control to the main program loop



Interrupt Enable Register



EA	IE.7	Disables all interrupts
--	IE.6	Not implemented, reserved for future use
ET2	IE.5	Enables or disables timer 2 overflow or capture interrupt (8952)
ES	IE.4	Enables or disables the serial port interrupt
ET1	IE.3	Enables or disables timer 1 overflow interrupt
EX1	IE.2	Enables or disables external interrupt 1
ET0	IE.1	Enables or disables timer 0 overflow interrupt
EX0	IE.0	Enables or disables external interrupt 0

CODE:

Program	Comments
<i>Toggling of p1.2 with the delay of 5ms with interrupts using timer 0</i>	
ORG 0000H	Initialization of the starting address
LJMP MAIN	Long jump to main program
ORG 000BH	ISR for Timer 0
CPL P1.2	Toggle or complement the state of P1.2
MOV TL0, #0FFH	Load calculated count for delay in Timer, lower byte in lower register and higher byte in higher register
MOV TH0, #0EDH	
RETI	Return from Interrupt
ORG 30H	Initialization of the main program address
MAIN: MOV TMOD,#01h	Timer 0, Mode 1
MOV TL0, #0FFh	
MOV TH0, #0EDH	
MOV IE, #82H;	Enable the global interrupt , enable Timer 0 interrupt
SETB TR0	
HERE: SJMP HERE	
END	



JSPM UNIVERSITY PUNE

Recognized by UGC u/s 2 (f) of UGC Act 1956 and enacted by the
State Government of Maharashtra - JSPM University Act, 2022 (Mah.IV of 2023)

RESULT:

Before Execution:

Registers

Register	Value
Regs	0x00
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00

Sys

Register	Value
a	0x00
b	0x00
sp	0x07
sp_max	0x07
dptr	0x0000
PC \$	C.0x0033
states	4
sec	0.00000434
psw	0x00

Disassembly

```
1 ORG 0000H
2 LJMP MAIN
3 ORG 000BH; ISR for Timer 0
4 CPL P1.2
5 MOV TLO, #0FFH
6 MOV TH0, #0EDH
7 RETI
8 ORG 30H
9 MAIN: MOV TMOD, #01h; Timer 0, Mode 1
10 MOV TLO, #0ffH
11 MOV TH0, #0EDH
12 MOV IE, #82H; enable Timer 0 interrupt
13 SETB TR0
14 HERE: SJMP HERE
15 END
16
```

Parallel Port 1

Port 1: P1: [0xFF] 7 Bits 0
Pins: [0xFF] 7 Bits 0

Timer/Counter 0

Mode: 1: 16 Bit Timer/Counter
Timer

TCON: 0x00 TMOD: 0x01
TH0: 0x00 TL0: 0x00
Control: T0 Pin TF0
Status: Stop
TR0 GATE INTO#

After Execution:

Registers

Register	Value
Regs	0x00
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00

Sys

Register	Value
a	0x00
b	0x00
sp	0x07
sp_max	0x09
dptr	0x0000
PC \$	C.0x003E
states	928453257
sec	1007.43625977
psw	0x00

Disassembly

```
1 ORG 0000H
2 LJMP MAIN
3 ORG 000BH; ISR for Timer 0
4 CPL P1.2
5 MOV TLO, #0FFH
6 MOV TH0, #0EDH
7 RETI
8 ORG 30H
9 MAIN: MOV TMOD, #01h; Timer 0, Mode 1
10 MOV TLO, #0ffH
11 MOV TH0, #0EDH
12 MOV IE, #82H; enable Timer 0 interrupt
13 SETB TR0
14 HERE: SJMP HERE
15 END
16
```

Logic Analyzer

p1: 0, d: 255

Selected Interrupt

EA IT0 IE0 EX0 Pr: 0

Interrupt System

Int Source	Vector	Mode	Req	Ena	Pr.
P3.2/Int0	0003H	0	0	0	0
Timer 0	0008H	0	0	1	0
P3.3/Int1	0013H	0	0	0	0
Timer 1	0018H	0	0	0	0
Serial Rcv.	0023H	0	0	0	0
Serial Xmt.	0023H	0	0	0	0

Timer/Counter 0

Mode: 1: 16 Bit Timer/Counter
Timer

TCON: 0x10 TMOD: 0x01
TH0: 0xFA TL0: 0x29
Control: T0 Pin TF0
Status: Run
TR0 GATE INTO#

Conclusion:



Experiment-5

AIM:

Count no. of pulses without interrupts using counter 0

SOFTWARE REQUIRED: Keil μvision

ALGORITHM:

Algorithm for Count no. of pulses using counter 0

Step 1: Start(Main Program)

Step 2: Initialization /Configure Timer 0 for Counter Mode (Mode 1, 16-bit counter)

Step 3: Configure the input pin to behave like external pulse source (P3.4 for Timer 0)

Step 4: Initialize the Timer 0 registers (TH0 and TL0) with an initial count zero.

Step 5: Start Timer 0 by Set the TR0 bit in the TCON register (TCON.4)

Step 6: Continuously read the values in TL0 and TH0 to get the current count of pulses.

Step 7: Display the Count on Port 1 and Port 2 and repeat it.

Step 8: END

CODE:

Program	Comments
Count no. of pulses without interrupts ORG 0000H MOV TMOD, #05h SETB P3.4 MOV TL0, #00H MOV TH0, #00H SETB TR0 BACK: MOV A, TL0 MOV P1, A MOV A, TH0 MOV P2, A SJMP BACK END	Set Timer 0 for Counter Mode 1 (16-bit counter) Set the P3.4 for external pulse source Load the initial count value into TH0 and TL0 to set the starting point Set the TR0 bit in the TCON register Move the lower value of count in A Display lower count on Port 1 Move the higher value of count in A Display higher count on Port 1 Repeat the loop

RESULT:



JSPM UNIVERSITY PUNE

Recognized by UGC u/s 2 (f) of UGC Act 1956 and enacted by the State Government of Maharashtra - JSPM University Act, 2022 (Mah.IV of 2023)

Before Execution:

The screenshot shows the Keil MDK-ARM IDE interface. The assembly code window displays the following code:

```
ORG 0000H
MOV TMOD, #05h
SETB P3.4
MOV TL0, #00H
MOV TH0, #00H
SETB TR0
BACK: MOV A, TL0
MOV P1, A
MOV A, TH0
MOV P2, A
CPL P3.4
SJMP BACK
END
```

Configuration windows for Timer/Counter 0 and Parallel Ports are open. The Timer/Counter 0 window shows Mode set to "1: 16 Bit Timer/Counter" and Counter selected. The Parallel Port 3 window shows Port 3 configured with P3: 0xFF, 7 Bits, and 0 pins. The Parallel Port 1 window shows Port 1 configured with P1: 0xFF, 7 Bits, and 0 pins. The Parallel Port 2 window shows Port 2 configured with P2: 0xFF, 7 Bits, and 0 pins.

After Execution:

The screenshot shows the Keil MDK-ARM IDE interface after execution. The assembly code window remains the same as before. Configuration windows for Timer/Counter 0 and Parallel Ports are still open. The Timer/Counter 0 window now shows Status: Run. The Parallel Port 3 window shows Port 3 configured with P3: 0xEF, 7 Bits, and 0 pins. The Parallel Port 1 window shows Port 1 configured with P1: 0x03, 7 Bits, and 0 pins. The Parallel Port 2 window shows Port 2 configured with P2: 0x00, 7 Bits, and 0 pins.

Conclusion:



Experiment-6

AIM: Count no. of pulses with interrupts using counter 1

SOFTWARE REQUIRED: Keil μvision

ALGORITHM:

Algorithm for Count no. of pulses using counter 1 with an interrupt

Main program loop

Step 1: Start(Main Program)

Step 2: Initialization /Configure Timer 1 for Counter Mode (Mode 1, 16-bit counter):

Step 3: Configure the input pin by connecting it to the external pulse source (P3.5 for Counter 1)

Step 4: Initialize Timer 1 with a pre-count value: Load the TL1 and TH1 registers with 0FAH and 0FFH, respectively.

Step 5 : Enable Counter 1 interrupt in the IE register by loading it with 88H.

Step 6: Set the TR1 bit to start the counter and continuously set the output of Port 1 to 0AAH

Step 7: END

Counter 1 Interrupt Service Routine (ISR)

This routine is automatically executed when Counter 1 overflows (reaches its maximum count and resets to 0) and the interrupt is enabled:

Step 1: Select the address of Interrupt T1 at address 001BH

Step 2: Set the output of Port 1 to 55H to signal that the interrupt has been handled.

Step 3: Load the TL1 and TH1 registers with 0FAH and 0FFH again to prepare for the next counting cycle



Step 4: RETI instruction returns control to the main program loop

CODE:

Program	Comments
Count no. of pulses with interrupts using counter 1 ORG 0000H LJMP MAIN ORG 001BH MOV P1, #55H MOV TL1, #0FAH MOV TH1, #0FFH RETI ORG 30H MAIN: MOV TMOD, #50h SETB P3.5 MOV TL1, #0FAH MOV TH1, #0FFH MOV IE, #88H SETB TR1 HERE: MOV P1, #0AAH SJMP HERE END	Initialization of the starting address Long jump to main program ISR for Counter 1 Port 1 to 55H to signal that the interrupt has been handled. Load the count TL1 and TH1 registers with 0FAH and 0FFH, respectively Return from Interrupt Initialization of the main program address Counter 1, Mode 1 Pulse source (P3.5 for counter 1) is configured as an input Enable the global interrupt , enable Timer 1 interrupt Start the counter Port 1 to AAH to signal that the main program has been handled,

RESULT:

Before Execution:



JSPM UNIVERSITY PUNE

Recognized by UGC u/s 2 (f) of UGC Act 1956 and enacted by the State Government of Maharashtra - JSPM University Act, 2022 (Mah.IV of 2023)

The screenshot shows a debugger interface with the following components:

- File Edit View Project Flash Debug Peripherals Tools SVCS Window Help**: The menu bar.
- Registers**: Shows register values for r0 to r7 and various system variables (a, b, sp, etc.) in hex format.
- Disassembly**: Displays assembly code for E2d1.asm, including instructions like ORG, LJMP, MOV, SETB, CPL, RETI, and END.
- Logic Analyzer**: A window showing signal traces for Parallel Port 3 (Port 3) and Parallel Port 1 (Port 1).
- Interrupt System**: A table listing interrupt sources (P3.2/Int0, Timer 0, P3.3/Int1, Timer 1, Serial Rcv., Serial Xmt.) with their respective vector numbers (0003H, 0008H, 0013H, 0018H, 0023H, 0023H), modes (0, 0, 0, 0, 0, 0), and priorities (0, 0, 0, 0, 0, 0).
- Selected Interrupt**: A dropdown menu for selecting an interrupt source.
- Parallel Port 3**: A configuration dialog for Port 3, showing bits 7 to 0 set to 0xFF and pins also set to 0xFF.
- Parallel Port 1**: A configuration dialog for Port 1, showing bits 7 to 0 set to 0xFF and pins also set to 0xFF.
- Timer/Counter 1**: A configuration dialog for Timer/Counter 1, set to 1:16 Bit Timer/Counter mode, Counter mode, TCON: 0x00, TMOD: 0x50, TH1: 0x00, TL1: 0x00, and T1 Pin selected.

After Execution:

Main Program

The screenshot shows the debugger interface after the program has executed. The main differences are:

- Registers**: The PC register now shows the value 274684844.
- Disassembly**: The assembly code remains the same as in the previous screenshot.
- Logic Analyzer**: The signal traces for Parallel Port 3 and Parallel Port 1 show different patterns compared to the initial state.
- Interrupt System**: The table shows the same interrupt configuration as before.
- Selected Interrupt**: The dropdown menu is still present.
- Parallel Port 3**: The configuration dialog for Port 3 shows bits 7 to 0 set to 0xFF and pins set to 0xFF.
- Parallel Port 1**: The configuration dialog for Port 1 shows bits 7 to 0 set to 0xAA and pins set to 0xAA.
- Timer/Counter 1**: The configuration dialog for Timer/Counter 1 shows TCON: 0x40, TMOD: 0x50, TH1: 0xFF, TL1: 0xFA, and T1 Pin selected. The Status field is now set to Run.



JSPM UNIVERSITY PUNE

Recognized by UGC u/s 2 (f) of UGC Act 1956 and enacted by the
State Government of Maharashtra - JSPM University Act, 2022 (Mah.IV of 2023)

Interrupt Service Routine (ISR) - at address 001BH

The screenshot shows a debugger interface with the following components:

- File Edit View Project Flash Debug Peripherals Tools SVCS Window Help**: The menu bar.
- Registers**: Shows the state of various registers (r0-r7, Sys, PC, etc.) with values ranging from 0x00 to 0x55.
- Logic Analyzer**: A window showing signal traces for various pins over time.
- E2d1.asm**: The assembly code for the program:

```
1 ORG 0000H
2 LMP MAIN
3 ORG 001BH
4 MOV P1, #55H
5 MOV T1L, #0FAH
6 MOV TH1, #0FFH
7 RETI
8 ORG 30H
9 MAIN: MOV TMOD, #50h
10 SETB P3.5
11 MOV T1L, #0FAH
12 MOV TH1, #0FFH
13 MOV IE, #88H
14 SETB TR1
15 HERE: MOV P1, #0AAH
16 CPL P3.5
17 SJMP HERE
18 END
19
```
- Parallel Port 3**: Configuration for Port 3: P3: [0xDF] 7 Bits 0, Pins: [0xDF] 7 Bits 0.
- Parallel Port 1**: Configuration for Port 1: P1: [0x55] 7 Bits 0, Pins: [0x55] 7 Bits 0.
- Interrupt System**: A table showing interrupt sources, vectors, modes, and enable status:

Int Source	Vector	Mode	Req	Ena	Pri
P3.2/Int0	0003H	0	0	0	0
Timer 0	000BH	0	0	0	0
P3.3/Int1	0013H	0	0	1	0
Timer 1	001BH	0	0	0	0
Serial Rcv.	0023H	0	0	0	0
Serial Xmt.	0023H	0	0	0	0
- Selected Interrupt**: A dropdown menu for selecting the interrupt source (I0, I0#, I1, I1#, EX0, EX1).
- Timer/Counter 1**: Configuration for Timer/Counter 1:

Mode
1: 16 Bit Timer/Counter

TCON: [0x40] TMOD: [0x50]
TH1: [0xFF] TL1: [0xFA]
Control: T1 Pin, TF1
Status: Run
 TR1 GATE INT1#

Conclusion:



Experiment-7

AIM: Program to transfer data serially with baud rate of 9600

SOFTWARE REQUIRED: Keil μvision

ALGORITHM:

Algorithm for transferring data serially with a baud rate of 9600

Step 1: Start

Step 2: 1. Initialize Serial Communication and load the TMOD by 20H to configure Timer 1 in Mode 2 (8-bit auto-reload mode).

Step 3: Loads the TH1 register with the value -3 (equivalent to 0xFD in hexadecimal) to generate the baud rate (9600) for serial communication.

Step 4: Load 50H in the Serial Control (SCON) register to work in Serial Mode 1 (8-bit UART with variable baud rate).

Step 5: Start Timer 1 to begin the baud rate generation by setting TR1 1

Step 6: Transmit the character that is written in the SBUF Register

Step 7: Wait for Data Transmission by continuously checking the TI (Transmit Interrupt) flag in the SCON register.

Step 8: TI flag is cleared by the program to prepare for the next character transfer

Step 9: Repeat for Continuous Transmission

CODE:

Program	Comments
<p>To transfer data serially with baud rate of 9600</p> <pre>MOV TMOD, #20H MOV TH1, #-3 MOV SCON, #50H SETB TR1 AGAIN: MOV SBUF, #'A' HERE: JNB TI, HERE</pre>	<p>Configure Timer 1 in Mode 2 (8-bit auto-reload mode).</p> <p>TH1 register with the value -3 for baud rate 9600</p> <p>Serial Mode 1 (8-bit UART with variable baud rate)</p> <p>Start Timer</p> <p>Transmit the character 'A' that is written in the SBUF</p> <p>Continuously checking the TI for transmission completion</p>



CLR TI
SJMP AGAIN
END

Clear Transmit Interrupt
Repeat the transmission

RESULT:

Before Execution:

The screenshot shows the Keil MDK-ARM IDE interface. The assembly code in the editor is:

```
    3: MOV SCON, #50H
C:0x0006 759850 MOV     SCON(0x98),#0x50
    4: SETB TR1
C:0x0009 D28E SETB     TR1 (0x88.6)
—
1: MOV TMOD, #20H
2: MOV TH1, #-3
3: MOV SCON, #50H
4: SETB TR1
5: AGAIN: MOV SBUF, #'A'
6: HERE: JNB TI, HERE
7: CLR TI
8: SJMP AGAIN
9: END
10
```

The Serial Channel configuration window shows:

- Mode: 8-Bit var.Baudrate
- SCON: 0x50 SBUF: 0x00
- IRQ: REN
- Baudrate: 0

After Execution:

The screenshot shows the Keil MDK-ARM IDE interface after execution. The assembly code in the editor is:

```
    5: AGAIN: MOV SBUF, #'A'
C:0x000B 759941 MOV     SBUF(0x99),#0x41
6: HERE: JNB TI, HERE
C:0x000E 3099FD JNB     TI(0x98.1),HERE(C:000E)
—
1: MOV TMOD, #20H
2: MOV TH1, #-3
3: MOV SCON, #50H
4: SETB TR1
5: AGAIN: MOV SBUF, #'A'
6: HERE: JNB TI, HERE
7: CLR TI
8: SJMP AGAIN
9: END
10
```

The Serial Channel configuration window shows:

- Mode: 8-Bit var.Baudrate
- SCON: 0x50 SBUF: 0x41
- IRQ: REN
- Baudrate: 9600

The Timer/Counter 1 configuration window shows:

- Mode: 2:8 Bit auto-reload
- TCON: 0xC0 TMOD: 0x20
- TH1: 0xFD TL1: 0xFD
- Control: T1 Pin, Run, TR1 GATE, INT1#

Conclusion



Experiment-8

AIM: Program to receive data serially with baud rate of 9600

SOFTWARE REQUIRED: Keil µvision

ALGORITHM:

Algorithm for Program to receive data serially with baud rate of 9600

Step 1: Start

Step 2: 1. Initialize Serial Communication and load the TMOD by 20H to configure Timer 1 in Mode 2 (8-bit auto-reload mode).

Step 3: Loads the TH1 register with the value -3 (equivalent to 0xFD in hexadecimal) to generate the baud rate (9600) for serial communication.

Step 4: Load 50H in the Serial Control (SCON) register to work in Serial Mode 1 (8-bit UART with variable baud rate).

Step 5: Start Timer 1 to begin the baud rate generation by setting TR1 1

Step 6: Wait for Data Reception by continuously checking the RI (Receive Interrupt) flag in the SCON register

Step 7: Once RI is set, the received 8-bit data is moved from the Serial Buffer (SBUF) register to the Accumulator (A) and then display using P1.

Step 8: RI flag is cleared by the program to prepare for the reception of the next character

Step 9: Repeat for Continuous Reception

CODE:

Program	Comments
<i>Program to receive data serially with baud rate of 9600</i>	
ORG 0000h	
MOV TMOD, #20H	Configure Timer 1 in Mode 2 (8-bit auto-reload mode).
MOV TH1, #-3	TH1 register with the value -3 for baud rate 9600
MOV SCON, #50H	Serial Mode 1 (8-bit UART with variable baud rate
SETB TR1	Start Timer



JSPM UNIVERSITY PUNE

Recognized by UGC u/s 2 (f) of UGC Act 1956 and enacted by the State Government of Maharashtra - JSPM University Act, 2022 (Mah.IV of 2023)

HERE: JNB RI, HERE
MOV A, SBUF
MOV P1, A
CLR RI
SJMP HERE
END

Wait for Data Reception by continuously checking the RI
Move the data from the Serial Buffer (SBUF) register to the Accumulator (A) and then display using P1.
Clear Receive Interrupt
Repeat the transmission

RESULT:

Before Execution:

The screenshot shows the Keil MDK-ARM IDE interface. The assembly code in the E3a.asm file is as follows:

```
3: MOV TMOD, #20H
C:0x0000 3098FD JNB RI, HERE
4: SETB TR1
C:0x0009 D29E SETB TR1 (0x88.6)
—
1: MOV THOD, #20H
2: MOV TH1, #3
3: MOV SCON, #50H
4: SETB TR1
5: HERE: JNB RI, HERE
6: MOV A, SBUF
7: MOV P1, A
8: CLR RI
9: SJMP HERE
10 END
```

The Registers window shows initial values for all registers. The Serial Channel configuration dialog is open, set to 8-Bit Shift Register mode with SCON: 0x00 and SBUF: 0x00. The Timer/Counter 1 configuration dialog is also open, set to 8-Bit auto-reload mode with TCON: 0x00, TMOD: 0x20, TH1: 0xFD, TL1: 0x00, and T1 Pin selected.

After Execution:

The screenshot shows the Keil MDK-ARM IDE interface after the program has run. The assembly code in the E3a.asm file is now:

```
5: HERE: JNB RI, HERE
C:0x000B 3098FD JNB RI (0x98.0), HERE (C:000B)
6: MOV A, SBUF
C:0x000E E599 MOV A, SBUF (0x99)

1: MOV TMOD, #20H
2: MOV TH1, #3
3: MOV SCON, #50H
4: SETB TR1
5: HERE: JNB RI, HERE
6: MOV A, SBUF
7: MOV P1, A
8: CLR RI
9: SJMP HERE
10 END
```

The Registers window shows updated values for the registers. The Serial Channel configuration dialog is open, set to 8-Bit var. Baudrate mode with SCON: 0x52 and SBUF: 0x42. The Timer/Counter 1 configuration dialog is also open, set to 8-Bit auto-reload mode with TCON: 0xC0, TMOD: 0x20, TH1: 0xFD, TL1: 0xFD, T1 Pin selected, and Run status indicated.

Conclusion :



Experiment-9

AIM: Sequence Generator Using Serial Interface in 8051.

SOFTWARE REQUIRED: Keil µvision

ALGORITHM:

Algorithm for Sequence Generator

CODE:

Program	Comments
Sequence Generator	

TX_CHAR EQU 175DH
ORG 0000H
LJMP MAIN_START
ORG 9000H
MAIN_START: MOV R4, #08
MOV A, #01H
NXT_CH: MOV R6, A
MOV B, A
ADD A, #30H
CJNE A, #3AH, N00
N00: JC N05
ADD A, #07H
N05: MOV SBUF, A
LCALL UART_TX_CHAR
MOV A, B
ADD A, #02H
DJNZ R4, NXT_CH
SJMP \$
ORG 175DH
UART_TX_CHAR: JNB TI, \$
CLR TI
RET
END

Define the subroutine address to send a character
Origin of Program
Jump to main program
Program starts at memory address 9000H
Initialize loop counter R4 with 8
Load A with 1 (initial number to convert and send)
Store current value of A into R6 (preserve original value)
Store current value of A in B also
Convert the number in A to ASCII by adding 0x30
Compare A with '9'+1 (0x3A); if not equal, jump to N00
If A < 0x3A (i.e., still a digit), jump to N05
Else (A was a letter A–F), adjust to get correct hex letter
Store the result in SBUF
Call the subroutine to transmit the character
Restore original value from B to A
Increment by 2 for next iteration
Decrement R4, loop until 8 iterations complete
End program (infinite loop)

UART Transmit Subroutine (Wait for TI flag)
Clear transmit interrupt flag
Return from ISR

RESULT:



JSPM UNIVERSITY PUNE

Recognized by UGC u/s 2 (f) of UGC Act 1956 and enacted by the
State Government of Maharashtra - JSPM University Act, 2022 (Mah.IV of 2023)

Before Execution:

The screenshot shows the Keil MDK-ARM IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The toolbar below has various icons for file operations and debugging. The left sidebar displays the 'Registers' window, which lists the CPU registers (r0-r7, sp, PC, etc.) with their current values. The main workspace contains the 'Disassembly' view, showing the assembly code for the program, and the 'E3c.asm' source code window. The assembly code starts at address C:0x0000 with the instruction LJMP MAIN_START (C:9000). The source code defines TX_CHAR as 175DH and contains the main loop logic. The bottom status bar shows the command line and the UART #1 terminal window.

After Execution:

The screenshot shows the Keil MDK-ARM IDE interface after the program has been executed. The assembly code in the disassembly view now starts at address C:0x0000 with the instruction LJMP MAIN_START (C:9000), indicating that the program has completed its execution. The source code window remains the same. The bottom status bar shows the command line and the UART #1 terminal window, which displays the output '13579BDF'.

Conclusion: