Date: 5/09/2025

5. PROGRAMS ON INTERRUPT MECHANISM

AIM: Write an assembly language program to generate a square wave with frequency of 2kHz and transfer the data from port 0 to port 1,when interrupt occurs.

TOOLS REQUIRED: PC, Keil µvision5

PROCEDURE:

1. Turn on the computer, create a folder on D drive saved with Register Number.
2. Open Keil uVision5 in desktop, or windows start menu -> all programs->open Keil uVision5.

Creating Project:

3. Go to project ->click on new uVision project -> create a new folder saved with experiment number within the already existed register number folder in D drive mentioned in step 1, -> enter the project name -> click on save.
4. Select the device for target -> In devices ->Enter P89C51RD2XX in Search toolbar -> click on ok -> select **No** for dialog box message "Copy STARTUP.A51 to project folder and add files to project".

(or)

Choose NXP -> to select the device P89C51RD2xx -> click on ok -> select **No** for Copy STARTUP.A51 to project folder.

Creating Coding File:

5. Go to file -> click on new->go to save (choose the path to save the file, It is saved within the name of experiment number folder mentioned in step 3)-> enter a file name with extension **.asm** ->save the file.

Linking the Coding File to Project :

6. Right-click on Source group1 in project bar-> Add existing files to source group1-> choose the experiment number folder path and select all files in the folder -> select .asm code file ->click on add-> click on close.
7. Write the assembly language program in .asm code file and save it.

Executing the Code File:

8. Right-click on .asm code file->Click on Build target to check the errors (i.e 0-Errors,0-Warning)
9. Go to debug->Click on Start/Stop Debug Session -> click on ok for dialog box message "running code size limit 2K" -> and Click on RUN in debug label
10. Observe the output in Register windows, Memory windows, Serial window.

CALCULATIONS:

Frequency of square wave: f = 2 kHz

• Period of square wave:

T = 1/f = 1/2000 = 500 μs

• Half-period (for high or low portion):

T/2 = 250 μs

• Single clock period: 1.085 μs

• Required delay=number of clocksXsingle clock period

250 μs=number of clocksX1.085 μs

Number of clocks=230

•Timer 1 in Mode 1 is a 16-bit timer:

Max clocks = 65536

• Initial count value:

65536 − 230 = 65306

• Hexadecimal representation:

65306 = FF1AH

• Timer register values:

TH0 = FFH, TL0 = 1AH

THEORY:

The IE register is an 8-bit special function register (SFR) used to enable or disable interrupts in the 8051.

D7							D0
EA	--	ET2	ES	ET1	EX1	ET0	EX0

EA	IE.7	Disables all interrupts.
--	IE.6	No implemented, reserved for future use
ET2	IE.5	Enables or disables timer 2 overflow interrupt
ES	IE.4	Enables or disables the serial port interrupt
ET1	IE.3	Enables or disables timer 2 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

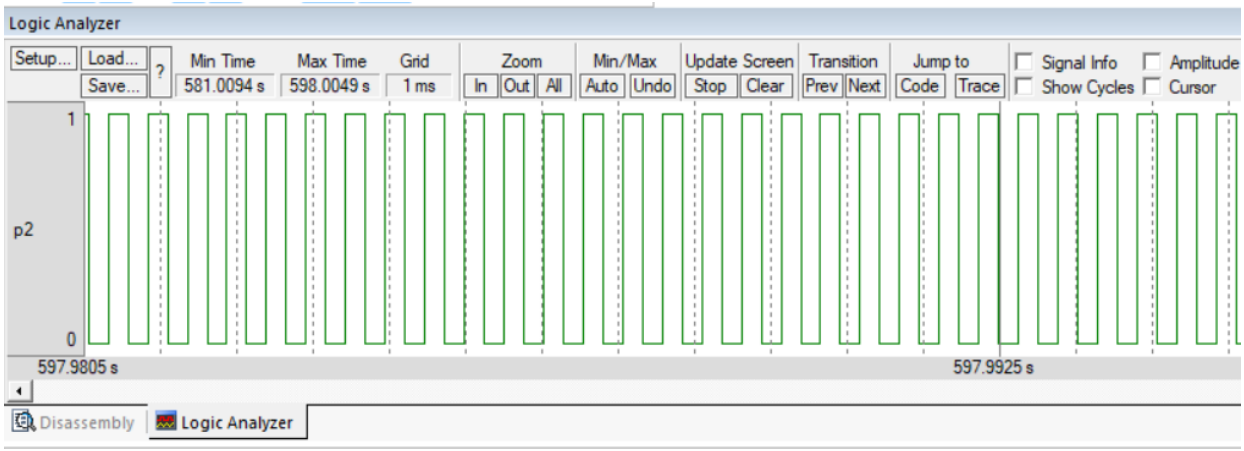
Interrupt vector table

Interrupt	ROM Location (hex)	Pin
Reset	0000	9
External HW (INT0)	0003	P3.2 (12)
Timer 0 (TF0)	000B	
External HW (INT1)	0013	P3.3 (13)
Timer 1 (TF1)	001B	
Serial COM (RI and TI)	0023	

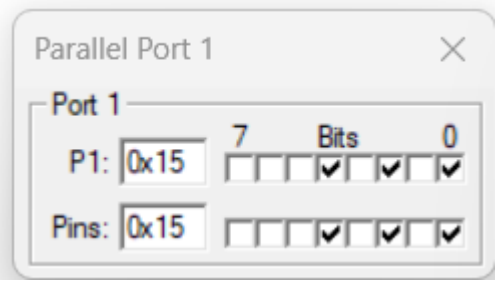
The 8051 microcontroller provides a versatile interrupt system that allows the processor to respond immediately to urgent events without continuously polling input devices. It supports five interrupts: External Interrupt 0 (INT0), Timer 0, External Interrupt 1 (INT1), Timer 1, and Serial Port Interrupt. Each interrupt has a unique vector address, where the processor jumps when the interrupt occurs. The IE (Interrupt Enable) register controls the activation of individual interrupts, while the EA (Enable All) bit acts as a global switch. When an interrupt is triggered, the processor completes the current instruction, saves the program counter on the stack, and executes the corresponding Interrupt Service Routine (ISR). After servicing, the processor returns to the main program using the RETI instruction. Interrupts in 8051 can also be assigned priority levels using the IP (Interrupt Priority) register, ensuring that more critical tasks are serviced first. This system allows the microcontroller to handle real-time events efficiently, making it suitable for applications like timers, serial communication, and external device control, where immediate response is essential.

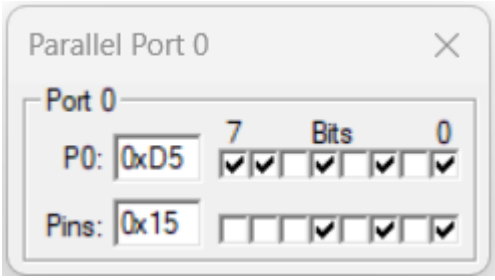
OUTPUT:

View→Analysis window→logic analyzer



Peripherals → $^{I}/_{O}$ Ports → Port 1 and port 0





PROGRAM:

Address	Opcode	LABELS	Mnemonic	Operand
0000	—		ORG	00H
0000	8062		SJMP	MAIN
0064	—		ORG	100
0064	758055		MOV	P0,#55H
0067	75A882		MOV	IE,#82H
006A	758901		MOV	TMOD,#01H
006D	758CFF		MOV	TH0,#0FFH
0070	758A1A		MOV	TL0,#1AH
0073	D28C		SETB	TR0
0075	E580		MOV	P0
0077	F590		MOV	P1,A
0079	80FA		SJMP	NEXT
000B	—		ORG	0BH
000B	B2A2		CPL	P2.2
000D	C28C		CLR	TR0
000F	758CFF		MOV	TH0,#0FFH
0012	758A1A		MOV	TL0,#1AH
0015	D28C		SETB	TR0
0017	32		RETI	
			END	

RESULT:

The assembly language program to generate a square wave with frequency of 2kHz

and transfer the data from port 0 to port 1,when interrupt occurs was successfully completed.