Experiment -5

BCD Addition

Program 1

Write a program to add 10 bytes of data and store the result in registers R2 and R3. The bytes are stored in the ROM space starting at 200H. The data would look as follows:
MYDATA: DB 92,34,84,129,... ;pick your own data.
Notice that you must first bring the data from ROM space into the CPU's RAM, then add them together. Use a simulator to single-step the program and examine the data.

```
ORG 000H
MOV DPTR,#200H; Initialize source pointer
MOV R0,#10; No of data
LOOP:CLR A
MOVC A,@A+DPTR; Fetch the data from ROM
ADD A,R2
JNC NEXT; Check for carry after addition
INC R3
NEXT:INC DPTR; Increment the source pointer
MOV R2,A ; Move the result to R2 register
DJNZ R0,LOOP ; check the count, if it is not zero continue with next addition
HERE:SJMP HERE
ORG 200H
DB 22H,43H,23H,34H,31H,77H,91H,33H,43H,7H
END
```

Program 2

Write a program to add 10 bytes of BCD data and store the result in R2 and R3. The bytes are stored in ROM space starting at 300H. The data would look as follows:
MYDATA: DB 92H,34H,84H,29H,... ;pick your own data.
Notice that you must first bring the data from ROM space into the CPU's RAM, then add them together. Use a simulator to single-step the program and examine the data.

```
ORG 000H
MOV DPTR,#300H; Initialize source pointer
MOV R0,#10H; No of data
LOOP:CLR A
MOVC A,@A+DPTR ; Fetch the data from ROM
ADD A,R2
DA A; Decimal adjust accumulator
JNC NEXT ; Check for carry after addition
INC R3
```

```
NEXT:INC DPTR ; Increment the source pointer
MOV R2,A ; Move the result to R2 register
DJNZ R0,LOOP ; check the count, if it is not zero continue with next addition
HERE:SJMP HERE
ORG 300H
DB 22H,43H,23H,34H,31H,77H,91H,33H,43H,7H
END
```

Program 3

Write a program to add two multi-byte BCD numbers together and store the result in RAM locations 40H - 44H. The two multi-byte items are stored in the ROM space starting at 120H and 150H. See the following example data.

```
            ORG  120H
DATA_1:     DB     54H,76H,65H,98H     ;number 98657654H
DATA_2      DB     93H,56H,77H,38H     ;number 38775693H
```

Pick your own data for your program. Notice that you must first bring the data from ROM space into the CPU's RAM and then add them together. Use a simulator to single-step the program and examine the data.

```
ORG 0000H
MOV R2,#4H; INTIALIZE COUNT TO ADD 4 TIMES
MOV R0,#40H ; INITALIZE DESTINATION POINTER
MOV DPTR,#120H ; INITALIZE SOURCE POINTER
LOOP:CLR A
MOVC A,@A+DPTR ; FETCH A BYTE FROM THE MULTIBYTE DATA AVAILABLE IN ROM
MOV R1,A
MOV A,#30H;
MOVC A,@A+DPTR; FETCH A BYTE FROM THE MULTIBYTE DATA AVAILABLE IN ROM
ADDC A,R1; ADD WITH CARRY
DA A; DECIMAL ADJUST ACCUMULATOR
MOV @R0,A; MOVE THE RESULT TO RAM
INC R0
INC DPTR
DJNZ R2,LOOP; CHECK FOR COUNT
HERE: SJMP HERE
ORG 120H
DB 54H,76H,65H,98H
ORG 150H
DB 93H,56H,77H,38H
END
```

Experiment -6

Timer and Counter Programming

Program 1

Write a program using timer 0 to generate a 500 Hz square wave frequency on one of the pins of P1. Then examine the frequency using the KEIL IDE inbuilt Logic Analyzer.

Initial value calculation :

Frequency f=500Hz
Total time period T=2ms
Required delay=TON=TOFF=1ms
[(FFFF-XXXX)+1]*1.085µs = 1ms
(65536-X)* 1.085µs = 1ms
X=(64614)D
X=XXXX = FC66H

```
        ORG  0000H
        MOV TMOD,#01H
HERE:   MOV TL0,#66H
        MOV TH0,#0FCH
        CPL P1.0
        ACALL DELAY
        SJMP HERE
DELAY:  SETB TR0
AGAIN:  JNB TF0, AGAIN
        CLR TR0
        CLR TF0
        RET
        END
```

Program 2

Write a program using timer 1 to generate a 1 kHz square wave frequency on one of the pins of P1. Then examine the frequency using the oscilloscope.

Initial value calculation:
Frequency f=1 KHz
Total time period T=1ms
Required delay=TON=TOFF=0.5ms
[(FFFF-XXXX)+1]*1.085µs = 0.5ms
XXXX = FE33H

```
        ORG  0000H
```

```
        MOV TMOD,#10H
HERE:   MOV TL1,#33H
        MOV TH1,#0FEH
        CPL P1.0
        ACALL DELAY
        SJMP HERE
DELAY:  SETB TR1
AGAIN:  JNB TF1,AGAIN
        CLR TR1
        CLR TF1
        RET
        END
```

Experiment -7

Counter Programming

Program 1

Assuming that external clock pulses are fed into pin P3.5, write a program for counter 1 in mode 2 to count the pulses and display the TL1 count on P2, which connects to 8 LEDs.

```
MOV TMOD, #01100000B      ;counter 1, mode 2 (8-bit auto reload.); C/T'=1
MOV TH1, #0               ;clear TH1 (will be auto reloaded in to TL1 and
TL1 starts counting)
SETB P3.5                 ;make T1 pin as input
AGAIN: SETB TR1           ;start the counter
BACK: MOV A, TL1          ;get copy of TL
MOV P2, A                 ;display it on port 2
JNB TF1, BACK             ;keep doing, if TF ≠ 1 (no over flow/roll over)
CLR TR1                   ;stop the counter 1
CLR TF1                   ;make TF=0
SJMP AGAIN                ;keep doing it
```

Program 2

Assume a room can accommodate 14 people. Write an 8051 program to count the number of people entering the room. If the count reaches 14 turn on LED.

Initial value calculation :

```
ORG  0000H
MOV TMOD,#06h;  counter 0 in mode2
CLR P0.1;TURN OFF LED
SETB P3.4;   intilalize i/p port
MOV TH0,#00
SETB TR0;    Start Counter0
AGAIN:MOV  A, TL0
MOV P1,A
CJNE A,#10D ,AGAIN
SETB P0.1; turn ON LED.
END
```

Program 3

Design a counter for counting the pulses of an input signal for 1 second. The pulses to the counter one fed to pin 3.4 (Cristal Oscillator Frequency= 22MHz)

Experiment -11A

Timer Interrupt

Program 1

- Write an 8051 program to get data from one port and send it to another port continuously while an interrupt will do the following: One of the timers will toggle the P2.1 bit every 100 microseconds.

Initial value calculation:

Time delay is 100 microseconds
Timer0 in mode 2
[(FFH-XX)+1] X 1.085µs=100µs
XX=A3H

```
        ORG 0000H
        LJMP MAIN

        ORG 000BH
        CPL P2.1
        RETI

        ORG 0030H
MAIN: MOV TMOD,#02H
        MOV P0,#0FFH
        MOV TH0,#A3h
        MOV IE,#10000010B
        SETB TR0
BACK: MOV A,P0
        MOV P1,A
        SJMP BACK
        END
```

Experiment -11B

Serial Interrupt

Program 1

Write an 8051 program to get data from a single bit of P1.2 and send it to P1.7 continuously while an interrupt will do the following: A serial interrupt service routine will receive data from a PC and display it on one of the ports.

```
        ORG 0000H
        LJMP MAIN
        ORG 0023H
        LJMP SERIAL
        ORG 0030H
 MAIN:SETB P1.2
        MOV TMOD,#20H
        MOV TH1,#-3
        MOV SCON, #50H
        MOV IE,#10010000B
        SETB TR1
BACK:MOV C,P1.2
        MOV P1.7,C
        SJMP BACK
SERIAL:JB TI,TRANS
        MOV A,SBUF
        MOV P2,A
        CLR RI
        RETI
        TRANS:CLR TI
        RETI
        END
```

```
ORG   0H
START:
ACALL LCDINIT
ACALL GETKEY
AGAIN:        SJMP AGAIN
LCDINIT:      MOV   A,#38H          ;INIT. LCD 2 LINES, 5X7 MATRIX
ACALL COMNWRT                       ;call command subroutine
ACALL DELAY                 ;give LCD some time
MOV   A,#0EH                ;display on, cursor on
ACALL COMNWRT                       ;call command subroutine
ACALL DELAY                 ;give LCD some time
MOV   A,#01           ;clear LCD
ACALL COMNWRT                       ;call command subroutine
ACALL DELAY                 ;give LCD some time
MOV   A,#06H                ;shift cursor right
ACALL COMNWRT                       ;call command subroutine
ACALL DELAY                 ;give LCD some time
MOV   A,#80H                ;cursor at line 1, pos. 4
ACALL COMNWRT                       ;call command subroutine
ACALL DELAY                 ;give LCD some time
MOV   A,#'K'          ;display letter K
ACALL DATAWRT               ;call display subroutine
ACALL DELAY                 ;give LCD some time
MOV   A,#'E'          ;display letter E
ACALL DATAWRT               ;call display subroutine
ACALL DELAY                 ;give LCD some time
MOV   A,#'Y'          ;display letter Y
ACALL DATAWRT               ;call display subroutine
ACALL DELAY                 ;give LCD some time
MOV   A,#':'          ;display letter :
ACALL DATAWRT               ;call display subroutine
ACALL DELAY                 ;give LCD some time
RET
COMNWRT:MOV   P1,A
CLR   P2.0
CLR   P2.1
SETB  P2.2
ACALL DELAY
CLR   P2.2
RET
DATAWRT:MOV   P1,A
SETB  P2.0
CLR   P2.1
SETB  P2.2
ACALL DELAY
CLR   P2.2
RET
DELAY:MOV   R3,#50            ;50 or higher for fast CPUs
HERE2:MOV   R4,#255           ;R4 = 255
HERE:DJNZ  R4,HERE            ;stay until R4 becomes 0
DJNZ  R3,HERE2
RET
```

```
                ;Keyboard subroutine. This program sends the ASCII
                ;Code for pressed key to P0.1
                ;P0.4-P0.7 connected to rows, P0.0-P0.3 to column

GETKEY:     MOV P0,#0FH         ;----- to declare p0.0-p0.3 i/p
K1:MOV P0,#0FH
MOV A,P0
ANL A,#0FH
MOV P0,A
MOV A,P0
ANL A,#0FH
CJNE A,#0FH,K1              ;---------- To check any key pressed already
K2:ACALL DELAY
MOV A,P0
ANL A,#0FH
CJNE A,#0FH,OVER
SJMP K2
OVER:ACALL DELAY
MOV A,P0
ANL A,#0FH
CJNE A,#0FH,OVER1
SJMP K2
OVER1:CLR  P0.4             ;ROW1 SELECTED
SETB P0.5
SETB P0.6
SETB P0.7
MOV A,P0
ANL A,#0FH
CJNE A,#0FH,ROW0
CLR P0.5                    ;ROW2 SELECTED
SETB P0.7
SETB P0.6
SETB P0.4
MOV A,P0
ANL A,#0FH
CJNE A,#0FH,ROW1
CLR P0.6                    ;ROW3 SELECTED
SETB P0.7
SETB P0.5
SETB P0.4
MOV A,P0
ANL A,#0FH
CJNE A,#0FH,ROW2
CLR P0.7                    ;ROW4 SELECTED
SETB P0.4
SETB P0.6
SETB P0.5
MOV A,P0
ANL A,#0FH
CJNE A,#0FH,ROW3
SJMP K2
MOV R0,#04H
```

```
ROW0:MOV DPTR,#KCODE0
SJMP FIND
ROW1:MOV DPTR,#KCODE1
SJMP FIND
ROW2:MOV DPTR,#KCODE2
SJMP FIND
ROW3:MOV DPTR,#KCODE3
FIND:RRC A
JNC MATCH
INC DPTR
DJNZ R0,FIND
MATCH:MOV   A,#84H       ;display pressed key
ACALL COMNWRT            ;
ACALL DELAY
CLR  A                   ;set A=0 (match is found)
MOVC A,@A+DPTR           ;get ASCII from table
ACALL DATAWRT            ;call display subroutine
ACALL DELAY              ;give LCD some time
LJMP K1                  ;ASCII LOOK-UP TABLE FOR EACH ROW
ORG  300H
KCODE0: DB   'F','B','8','4' ;ROW 0
KCODE1: DB   'E','A','7','3' ;ROW 1
KCODE2: DB   'D','0','6','2' ;ROW 2
KCODE3: DB   'C','9','5','1' ;ROW 3
END
```