INDIAN INSTITUTE OF TECHNOLOGY
KHARAGPUR
DEPARTMENT OF ELECTRONICS AND ELECTRICAL COMMUNICATION

EC49001

MICROCONTROLLERS LABORATORY

ASSIGNMENT 5

SHIKHAR MOHAN

18EC10054
E & ECE

B. Tech.
2021 – 2022

Question:

1. From the interface, send your roll number in text format to 8051 and store the roll number in bytes starting from 30H.
2. Transmit the roll number from 8051 to UART interface and display it in the same.

Objectives:

To familiarise ourselves with how communication takes place between microcontrollers. In this case, we use the serial communication protocol UART.

Description:

i) UART:
The above, standing for Universal Asynchronous Receiver/Transmitter, is a form of serial communication protocol which interfaces with sending and receiving data over just 2 GPIO lines.

ii) Configuration:
The receiver and transmitter both have a baud rate of 4800 and we set UART config by setting individual bits of the SCON register; namely we clear SM0 and set SM1 and REN bits to enable serial mode in 8-bit UART mode and the serial mode receiver as well. Baud rate is set using timer 1, the autoreload function of which resets the timer after overflowing to make the timing activity of our UART communication perfect.

Code:

```
ORG 0000H
CLR SM0
SETB SM1; Serial mode in 8-bit UART mode
SETB REN; Serial port receiver enabled

MOV PCON, #128

MOV TMOD, #20H
MOV TH1, #243; Calculation for this
MOV TL1, #243; Explained in report
SETB TR1
MOV R1, #30H; This is where storage begins
MOV R2, #9; Size of roll number

LOOP:; Loop for receiving roll number input
JNB RI, $; Wait for data to be received
CLR RI
MOV A, SBUF; Move input in buffer to ACC
MOV @R1, A; Store input to memory
```

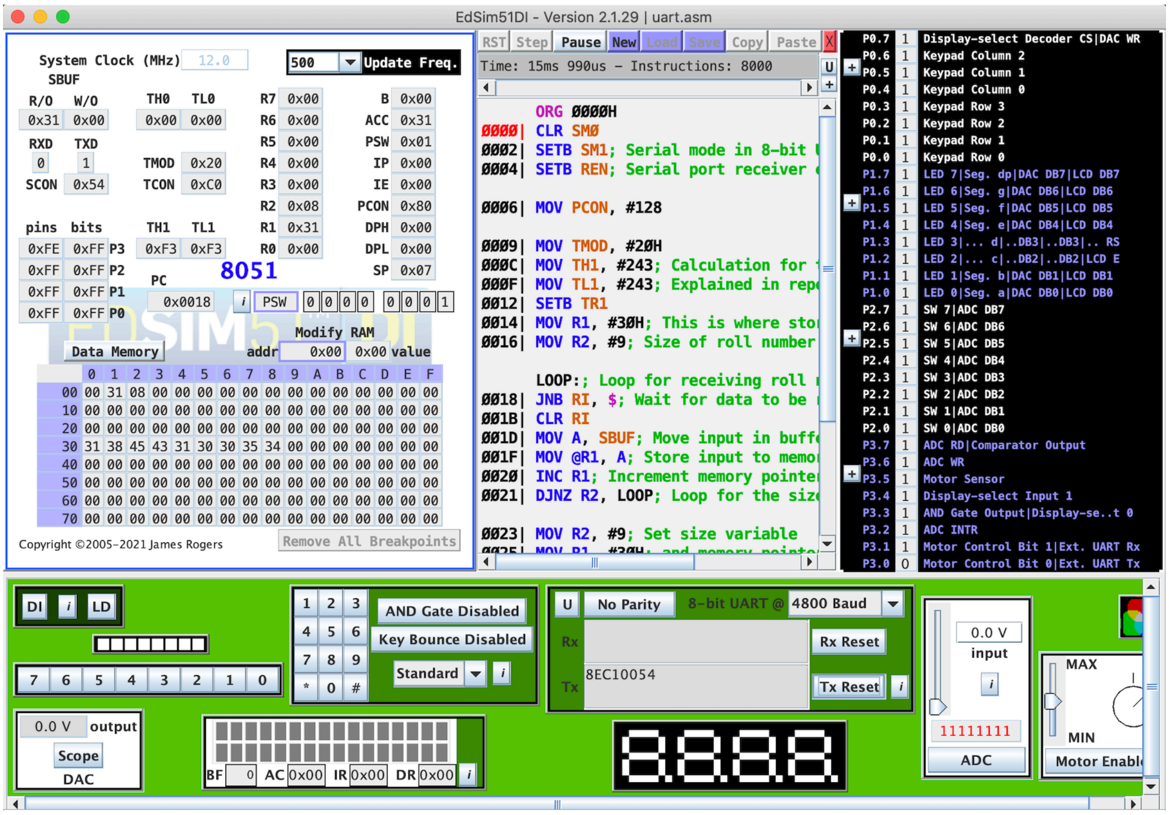
```
INC R1; Increment memory pointer
DJNZ R2, LOOP; Loop for the size of roll number

MOV R2, #9; Set size variable
MOV R1, #30H; and memory pointer again

LOOP2:; Loop for transmitting stored roll number
MOV SBUF, @R1; Move stored data to buffer
INC R1; Increment memory pointer
WAIT: JNB TI, WAIT; Wait for data to be transmitted
CLR TI
DJNZ R2, LOOP2; Loop for the size of roll number

```

Screenshot:


EdSim51DI - Version 2.1.29 | uart.asm

System Clock (MHz) 12.0 — 500 Update Freq.

SBUF R/O 0x31, W/O 0x00; RXD 0, TXD 1; SCON 0x54

TH0 0x00, TL0 0x00; TMOD 0x20; TCON 0xC0; TH1 0xF3, TL1 0xF3

R7 0x00, R6 0x00, R5 0x00, R4 0x00, R3 0x00, R2 0x08, R1 0x31, R0 0x00

B 0x00, ACC 0x31, PSW 0x01, IP 0x00, IE 0x00, PCON 0x80, DPH 0x00, DPL 0x00, SP 0x07

pins	bits	
0xFE	0xFF	P3
0xFF	0xFF	P2
0xFF	0xFF	P1
0xFF	0xFF	P0

8051 — PC 0x0018 — PSW 0 0 0 0 0 0 0 1

Modify RAM addr 0x00 0x00 value

Data Memory

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	31	08	00	00	00	00	00	00	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	31	38	45	43	31	30	30	35	34	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Copyright ©2005-2021 James Rogers

Time: 15ms 990us - Instructions: 8000

```
      ORG 0000H
0000| CLR SM0
0002| SETB SM1; Serial mode in 8-bit U
0004| SETB REN; Serial port receiver e

0006| MOV PCON, #128

0009| MOV TMOD, #20H
000C| MOV TH1, #243; Calculation for t
000F| MOV TL1, #243; Explained in repo
0012| SETB TR1
0014| MOV R1, #30H; This is where stor
0016| MOV R2, #9; Size of roll number

      LOOP:; Loop for receiving roll n
0018| JNB RI, $; Wait for data to be r
001B| CLR RI
001D| MOV A, SBUF; Move input in buffe
001F| MOV @R1, A; Store input to memor
0020| INC R1; Increment memory pointer
0021| DJNZ R2, LOOP; Loop for the size

0023| MOV R2, #9; Set size variable
```

P0.7	1	Display-select Decoder CS|DAC WR
P0.6	1	Keypad Column 2
P0.5	1	Keypad Column 1
P0.4	1	Keypad Column 0
P0.3	1	Keypad Row 3
P0.2	1	Keypad Row 2
P0.1	1	Keypad Row 1
P0.0	1	Keypad Row 0
P1.7	1	LED 7|Seg. dp|DAC DB7|LCD DB7
P1.6	1	LED 6|Seg. g|DAC DB6|LCD DB6
P1.5	1	LED 5|Seg. f|DAC DB5|LCD DB5
P1.4	1	LED 4|Seg. e|DAC DB4|LCD DB4
P1.3	1	LED 3|... d|..DB3|..DB3|.. RS
P1.2	1	LED 2|... c|..DB2|..DB2|LCD E
P1.1	1	LED 1|Seg. b|DAC DB1|LCD DB1
P1.0	1	LED 0|Seg. a|DAC DB0|LCD DB0
P2.7	1	SW 7|ADC DB7
P2.6	1	SW 6|ADC DB6
P2.5	1	SW 5|ADC DB5
P2.4	1	SW 4|ADC DB4
P2.3	1	SW 3|ADC DB3
P2.2	1	SW 2|ADC DB2
P2.1	1	SW 1|ADC DB1
P2.0	1	SW 0|ADC DB0
P3.7	1	ADC RD|Comparator Output
P3.6	1	ADC WR
P3.5	1	Motor Sensor
P3.4	1	Display-select Input 1
P3.3	1	AND Gate Output|Display-se..t 0
P3.2	1	ADC INTR
P3.1	1	Motor Control Bit 1|Ext. UART Rx
P3.0	0	Motor Control Bit 0|Ext. UART Tx

No Parity — 8-bit UART @ 4800 Baud

Rx:

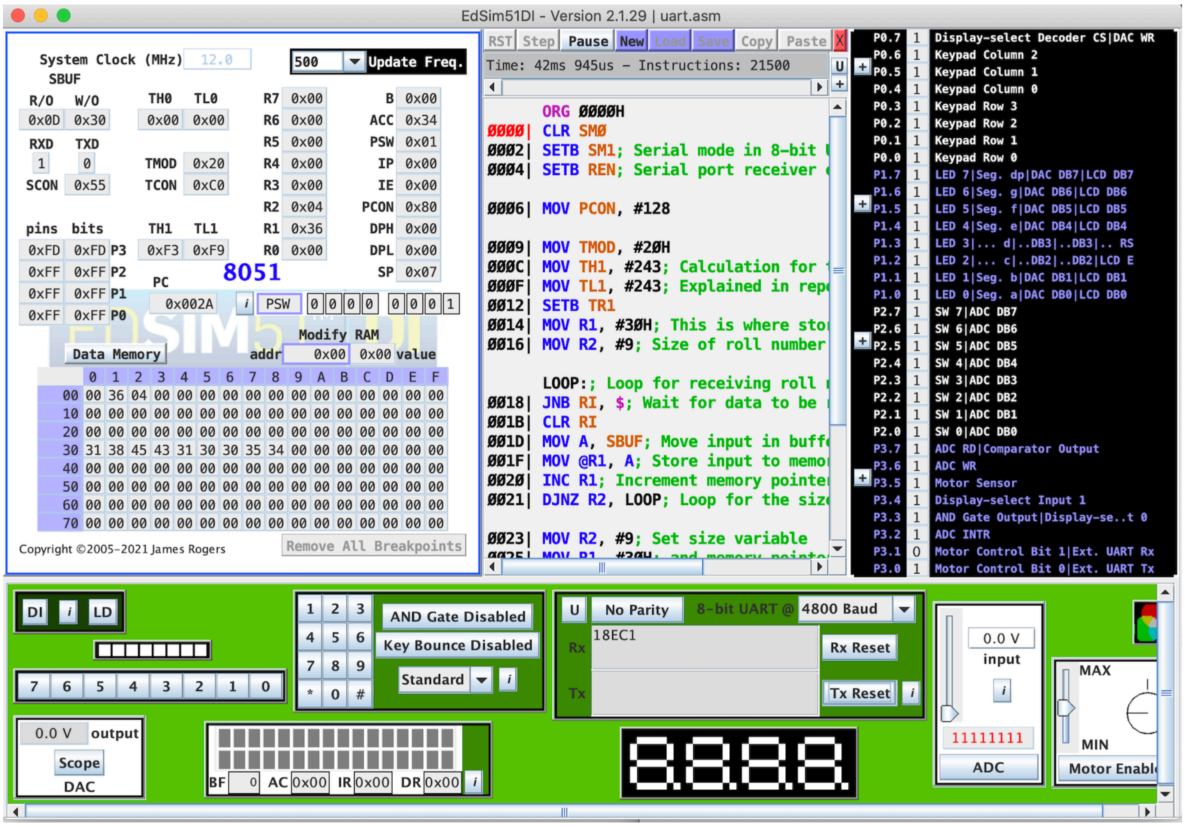
Tx: 8EC10054

AND Gate Disabled — Key Bounce Disabled — Standard

DAC: 0.0 V output — ADC: 0.0 V input, 11111111

BF 0 AC 0x00 IR 0x00 DR 0x00

Figure 1: Roll number data being received via UART. Note that memory locations 30H to 38H already have this data because of a previous run, but the data is still being transmitted.



EdSim51DI - Version 2.1.29 | uart.asm

```
System Clock (MHz) 12.0          500 Update Freq.
 SBUF
R/O   W/O      TH0   TL0      R7 0x00       B 0x00
0x0D  0x30     0x00  0x00     R6 0x00     ACC 0x34
                              R5 0x00     PSW 0x01
RXD   TXD      TMOD  0x20     R4 0x00      IP 0x00
 1     0       TCON  0xC0     R3 0x00      IE 0x00
SCON  0x55                    R2 0x04    PCON 0x80
pins  bits     TH1   TL1      R1 0x36     DPH 0x00
0xFD  0xFD P3  0xF3  0xF9     R0 0x00     DPL 0x00
0xFF  0xFF P2           8051               SP 0x07
0xFF  0xFF P1  PC
0xFF  0xFF P0  0x002A    PSW  0 0 0 0  0 0 0 1
                              Modify RAM
Data Memory              addr 0x00  0x00 value
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	36	04	00	00	00	00	00	00	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	31	38	45	43	31	30	30	35	34	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Copyright ©2005-2021 James Rogers

Time: 42ms 945us - Instructions: 21500

```
      ORG 0000H
0000| CLR SM0
0002| SETB SM1; Serial mode in 8-bit U
0004| SETB REN; Serial port receiver e

0006| MOV PCON, #128

0009| MOV TMOD, #20H
000C| MOV TH1, #243; Calculation for t
000F| MOV TL1, #243; Explained in repo
0012| SETB TR1
0014| MOV R1, #30H; This is where stor
0016| MOV R2, #9; Size of roll number

      LOOP:; Loop for receiving roll n
0018| JNB RI, $; Wait for data to be r
001B| CLR RI
001D| MOV A, SBUF; Move input in buffe
001F| MOV @R1, A; Store input to memor
0020| INC R1; Increment memory pointer
0021| DJNZ R2, LOOP; Loop for the size

0023| MOV R2, #9; Set size variable
0025| MOV R1, #30H; and memory pointe
```

P0.7	1	Display-select Decoder CS|DAC WR
P0.6	1	Keypad Column 2
P0.5	1	Keypad Column 1
P0.4	1	Keypad Column 0
P0.3	1	Keypad Row 3
P0.2	1	Keypad Row 2
P0.1	1	Keypad Row 1
P0.0	1	Keypad Row 0
P1.7	1	LED 7|Seg. dp|DAC DB7|LCD DB7
P1.6	1	LED 6|Seg. g|DAC DB6|LCD DB6
P1.5	1	LED 5|Seg. f|DAC DB5|LCD DB5
P1.4	1	LED 4|Seg. e|DAC DB4|LCD DB4
P1.3	1	LED 3|... d|..DB3|..DB3|.. RS
P1.2	1	LED 2|... c|..DB2|..DB2|LCD E
P1.1	1	LED 1|Seg. b|DAC DB1|LCD DB1
P1.0	1	LED 0|Seg. a|DAC DB0|LCD DB0
P2.7	1	SW 7|ADC DB7
P2.6	1	SW 6|ADC DB6
P2.5	1	SW 5|ADC DB5
P2.4	1	SW 4|ADC DB4
P2.3	1	SW 3|ADC DB3
P2.2	1	SW 2|ADC DB2
P2.1	1	SW 1|ADC DB1
P2.0	1	SW 0|ADC DB0
P3.7	1	ADC RD|Comparator Output
P3.6	1	ADC WR
P3.5	1	Motor Sensor
P3.4	1	Display-select Input 1
P3.3	1	AND Gate Output|Display-se..t 0
P3.2	1	ADC INTR
P3.1	0	Motor Control Bit 1|Ext. UART Rx
P3.0	1	Motor Control Bit 0|Ext. UART Tx

Rx: 18EC1

Figure 2: Roll number data is being transmitted via the UART to the interface.

Results:

We can clearly see that the transmitter is transmitting the roll number at a baud rate of 4800 bits/s. The receiving end is working just as well as we can observe the data get stored in memory locations 30H to 38H.

Discussion:

1. In this experiment we keep the clock frequency to 12MHz, which results in an occasional misprint since the exact baud rate of 4800 cannot be achieved with this number. This is acceptable for casual applications but not when sensitive information is to be transmitted.
2. We set the PCON.7 bit, which means we set our timers accordingly to send out triggers at the right moment by the following equation:

$$TH_1 = 256 - \frac{f_{\text{xtal}}}{192 * |\text{baud rate}|}$$

which gives us the value of nearly 243, which can be seen used in the code.

Summary:

It can be clearly seen that the steps are correct and our transmission and reception algorithm works perfectly, since we observe outputs as per our expectation.