LAB TITLE AND CODES EMBEDDED COMPUTING LAB (19CLE243)
EXPERIMENT NUMBER : 3
DATE : 26/04/2022 (TUESDAY)

* AIM :
To configure MSP43x on-chip peripherals and establish serial communication - transmission and reception.

* ALGORITHM : (CONTINUOUS TRANSMISSION OF A CHARACTER USING UART)
① Put in reset mode and disable oversampling.
② Set 8-bits data - NO Parity - 1 stop bit for Transmission (LSB)
③ In asynchronous mode, first LSB and then MSB should be set, followed by SMCLK.
④ Enabled EUSCI_A0 logic is held in reset state (SI).
⑤ Configure functionality of P1.2, P1.3 as UART pins.
⑥ Take UART out of reset mode.
⑦ In the main function after calling the UART transmission function, perform the following steps in an infinite loop -
(i) Wait until transmitter buffer is empty
(ii) Send a character
(iii) Delay

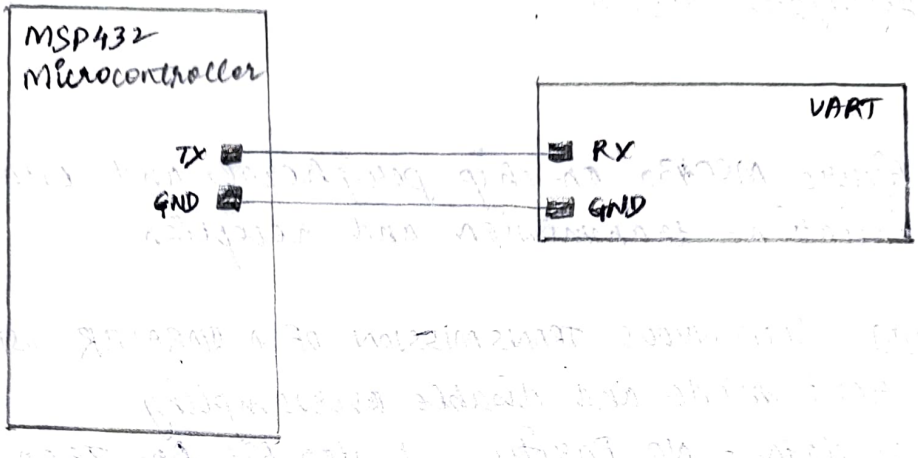
* SOURCE CODE : (CONTINUOUS TRANSMISSION OF A CHARACTER USING UART)

```
#include "msp.h"
void UART0_init(void); // Function for UART Transmission
void delayms(int n); // Delay Function

// main Function :
int main(void)
{
  UART0_init(); // Call UART Transmission Function
```

→ P1)

```
// Infinite Loop (An embedded program does not stop):
while (1)
{
  while (!(EUSCI_A0 -> IFG & 0x02)) {} // Wait until transmission
                                          buffer is empty
  EUSCI_A0 -> TXBUF = 'Y'; // Send a character

  while (!(EUSCI_A0 -> IFG & 0x02)) {} // Wait until transmission
                                          buffer is empty
  EUSCI_A0 -> TXBUF = 'E'; // Send a character

  while (!(EUSCI_A0 -> IFG & 0x02)) {} // Wait until transmission
                                          buffer is empty
  EUSCI_A0 -> TXBUF = 'S'; // Send a character

  delayMs(2); // Delay by 2 ms
}
}

// Function for UART Transmission:
void UART0_init(void)
{
  EUSCI_A0 -> CTLW0 |= 1; // Put in reset mode to configure UART
  EUSCI_A0 -> MCTLW = 0; // Disable oversampling
  EUSCI_A0 -> CTLW0 = 0x0081; // 00 - 1 stop bit for Transmission,
  No Parity, 8-bits data, Asynchronous mode, First LSB Then MSB,
  SMCLK, 81 - Enabled EUSCI_A0 Logic is held in reset state

  EUSCI_A0 -> BRW = 26; // 3000000/115200 = 26
  P1 -> SEL0 |= 0x0C; // Configure functionality of P1.2, P1.3 as
                         UART pins
  P1 -> SEL1 &= ~0x0C;
  EUSCI_A0 -> CTLW0 &= ~1; // Take UART out of reset mode
}
```

* OUTPUT (CONTINUOUS TRANSMISSION OF A CHARACTER USING UART) :



→ SERIAL WINDOW :-

```
YES YES YES YES...
```

```
// Delay milliseconds when system clock is at 8 MHz for Rev C MCU:
void delayMs (int n)
{
  int i, j;
  for (j=0; j<n; j++)
     for (i=750; i>0; i--); // Delay of 1ms
}
```

* **ALGORITHM:** (RECEPTION AND TRANSMISSION OF A CHARACTER USING UART)

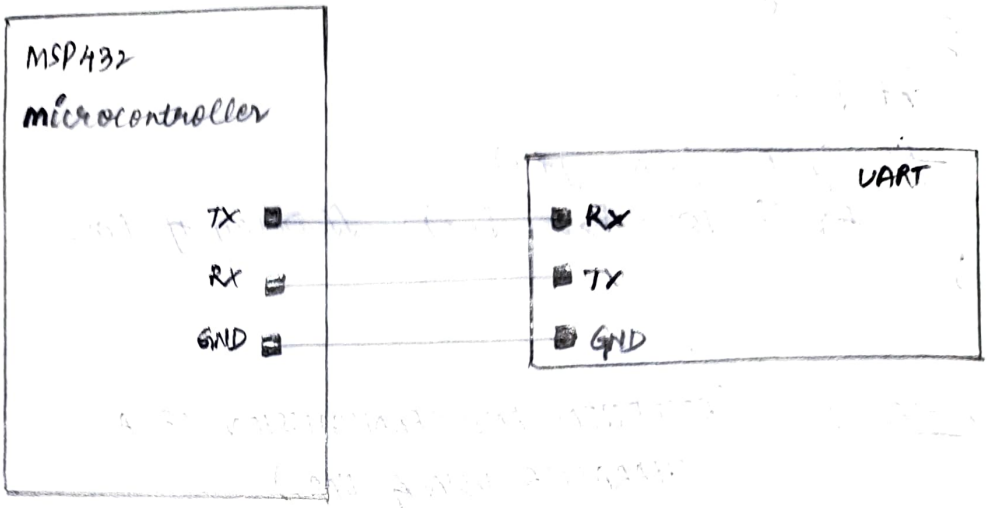
① Put in reset mode and disable oversampling.
② set 8-bits data - NO Parity - 1 stop bit for transmission. In asynchronous mode, first LSB and then MSB should be set, followed by SMCLK (00).
③ Enabled EUSCI_A0 logic is held in reset state (01).
④ Configure functionality of P1.2, P1.3 as UART pins.
⑤ Take UART out of reset mode.
⑥ In the main function after calling the UART transmission function, wait until transmitter buffer is empty and then, send the required number of characters bit-by-bit in a for loop.
⑦ In an infinite loop, perform the following steps -
(i) Wait until transmitter buffer is empty
(ii) Receive a character
(iii) Wait until transmitter buffer is empty
(iv) Display the character

* **SOURCE CODE:** (RECEPTION AND TRANSMISSION OF A CHARACTER USING UART)

```
#include "msp.h"
void UART0_init (void); // Function for UART Transmission
void delayMs (int n); // Delay Function

unsigned char c;
unsigned char message[] = "Ready\n";
int i;
```

```
// main function:
int main (void)
{
  UART0_init(); // call UART Transmission Function

  for (i=0; i<7; i++)
  {
    while (!(EUSCI_A0 -> IFG & 0x02)) {} // wait until transmitter
                                            buffer is empty
    EUSCI_A0 -> TXBUF = message[i]; // send a character
  }

  // Infinite loop (An embedded program does not stop):
  while (1)
  {
    while (!(EUSCI_A0 -> IFG & 0x02)) {} // wait until transmitter
                                            buffer is empty
    c = EUSCI_A0 -> RXBUF; // Receive a character

    while (!(EUSCI_A0 -> IFG & 0x02)) {} // wait until transmitter
                                            buffer is empty
    EUSCI_A0 -> RXBUF = c; // Display the character
  }
}

// Function for UART Transmission:
void UART0_init (void)
{
  EUSCI_A0 -> CTLW0 |= 1; // Put in reset mode to configure UART
  EUSCI_A0 -> MCTLW = 0; // Disable oversampling
  EUSCI_A0 -> CTLW0 = 0x0081; // 00 - 1 stop bit, NO Parity, 8-bits data,
  Asynchronous Mode, First LSB Then MSB, SMCLK (00), Enabled
  EUSCI_A0 logic is held in reset state (81)
```

* OUTPUT : (RECEPTION AND TRANSMISSION OF A CHARACTER USING UART)



→ SERIAL WINDOW :-

Ready

Receive a character : Amrita

Display the character : Amrita

```
    EUSCI_A0 → BRW = 26;  // 3000000 / 115200 = 26
    P1 → SEL0 |= 0x0C; // Configure functionality of P1.2, P1.3 as UART pins
    P1 → SEL1 &= ~0x0C;
    EUSCI_A0 → CTLW0 &= ~1; // Take UART out of reset mode
}

// Delay milliseconds when system clock is at 3MHz for Rev C MCU:
void delayMs (int n);
{
  int i, j;
  for (j=0; j<n; j++)
    for (i=750; i>0; i--); // Delay of 1 ms
}
```

* **ALGORITHM: (DEVICE CONTROL USING UART)**

① Put in reset mode and disable oversampling.

② Set 8-bits data - NO Parity - 1 stop bit for transmission. In asynchronous mode, first LSB and then MSB should be set, followed by SMCLK (00).

③ Enabled EUSCI_A0 logic is held in reset state (01).

④ Configure functionality of P1.2, P1.3 as UART pins.

⑤ Take UART out of reset mode.

⑥ In the main function after calling the UART transmission function, configure functionality of P2.1 as simple GPIO Port and configure direction of P2.1 as LED output.

⑦ In an infinite loop, perform the following steps -

(i) Wait until transmitter buffer is empty.

(ii) Receive a character

(iii) If character is A/a, turn ON P2.1 Green LED.

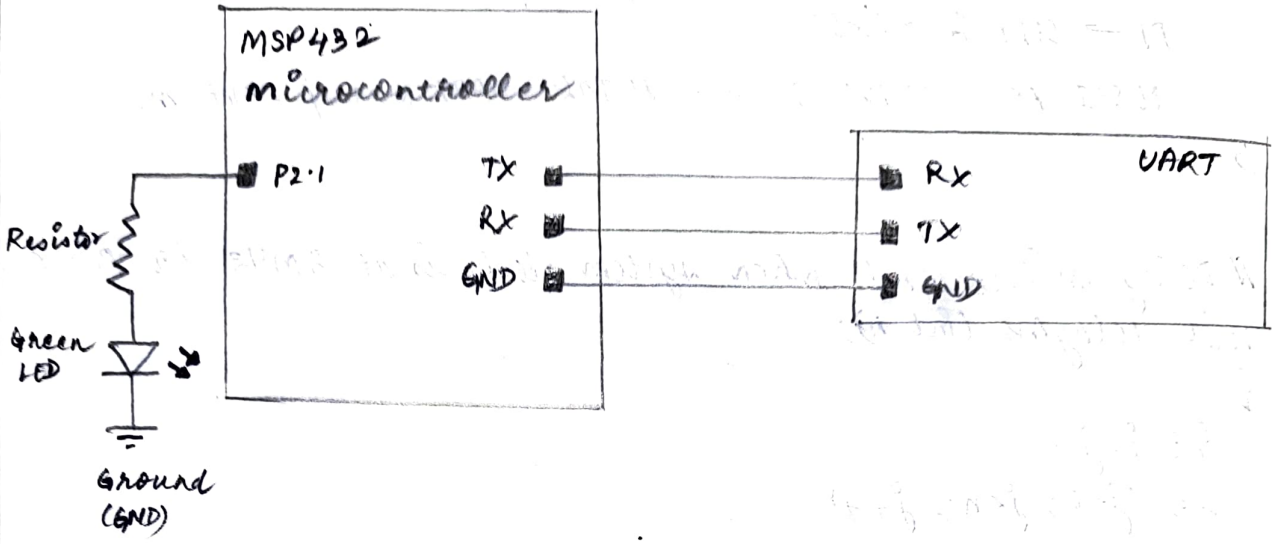
(iv) Else if character is B/b, turn OFF P2.1 Green LED.

PTO

* SOURCE CODE : (DEVICE CONTROL USING UART) :

```
#include "msp.h"
void UART0_init(void); //Function for UART Transmission
unsigned char c;

//Main Function:
int main(void)
{
  UART0_init(); // call UART Transmission Function

  P2->SEL1 &= ~2; // configure functionality of P2.1 as simple GPIO Port
  P2->SEL0 &= ~2;
  P2->DIR |= 2; // Configure direction of P2.1 as LED output

  // Infinite Loop (An embedded program does not stop):
  while(1)
  {
   while(!(EUSCI_A0->IFG & 0x01)) {} // wait until transmitter buffer is empty
   c = EUSCI_A0->RXBUF; // Receive a character

   if(c=='A' || c=='a')
   {
    P2->OUT |=2; // TURN ON P2.1 green LED
   }
   else if(c=='B' || c=='b')
   {
    P2->OUT &= ~2; // TURN OFF P2.1 Green LED
   }
  }
}
```

* OUTPUT : (DEVICE CONTROL USING UART)



→ SERIAL WINDOW :-

Receive a character: ABABAB

```
// Function for UART Transmission :
void UART0_init(void)
{
  EUSCI_A0 -> CTLW0 |= 1; // Put in reset mode to configure UART
  EUSCI_A0 -> MCTLW = 0; // Disable oversampling
  EUSCI_A0 -> CTLW0 = 0x0081; // 00- 1 stop bit, NO Parity, 8-bit data,
Asynchronous Mode, First LSB Then MSB, SMCLK ; 81- Enabled
EUSCI_A0 logic held in reset state
  EUSCI_A0 -> BRW = 26; // 3000000/115200 = 26
  P1 -> SEL0 |= 0x0C; // Configure functionality of P1.2, P1.3 as
UART pins
  P1 -> SEL1 &= ~0x0C;
  EUSCI_A0 -> CTLW0 &= ~1; // Take UART out of reset mode
}
```

* **RESULT :**

Configured MSP432 on-chip peripherals and established serial communications (transmission and reception). All simulation results were verified successfully.