ROBOTIC ARM INTERFACING WITH- MSP432P401R

TEAM MEMBERS

CB.EN. U4CCE21009 - ASHWANTHRAM A C

CB.EN. U4CCE21015 - DHANUSHRAJU S

CB.EN. U4CCE21020 - GAUTAM R S

CB.EN. U4CCE21026 - HARINI RAJ AKHTSHYA RG

B. TECH COMPUTER AND COMMUNICATION ENGINEERING

2021 BATCH

ROBOTIC ARM INTERFACING WITH- MSP432P401R

AIM:

To enable the MSP432 microcontroller to receive input commands via UART or button presses and use these commands to control the motor's movement. The code should allow for different modes of operation, such as automatic, UART, and switch mode, each triggering specific motor movements based on the received commands. The specific motor control actions will depend on the commands defined in the code and the associated pin configurations. These actions may include moving the motor forward, backward, or stopping its motion. The code should also handle cases where the motor needs to be turned off or where different motor control signals are required.

DESIGN:

PERIPHERALS AND ITS PIN USED DESCRIPTION:

PERIPHERAL	PINS USED	DESCRIPTION
UART	P1.2	UART communication pins for transmitting data
UART	P1.3	UART communication pins for receiving data.
GPIO	P2.0	To glow red LED.
GPIO	P2.1	To glow green LED.
GPIO	P2.2	To glow blue LED.
GPIO	P2.3	Control pin for the motor.
GPIO	P2.4	Control pin for the motor.
GPIO	P2.5	Control pin for the motor.
GPIO	P2.6	Control pin for the motor.
GPIO	P2.7	Control pin for the motor.
GPIO	P3.0	Input pin for Switch mode.
GPIO	P3.1	Input pin for Switch mode.
GPIO	P3.2	Input pin for Switch mode.
GPIO	P3.3	Input pin for Switch mode.
GPIO	P3.4	Input pin for Switch mode.
GPIO	P3.5	Input pin for Switch mode.
GPIO	P1.6	Input pin for Switch mode.
GPIO	P1.7	Input pin for Switch mode.

CODE:

```
#include "msp.h"
#include <string.h>

void UART0_init(void);
void delayMs(int n);
void ipconfig(void);


int main () {
int i;
unsigned char c;
  char message [] = "1. automatic,2. UART,3. switch\n";
char automatic [] = "automatic mode";

        char uart[] = "UART mode";
char button [] = "switch mode";
pinconfig();
  UART0_init (); // Call UART Transmission Function


   for (i = 0; i < strlen(message); 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 & 0x01)) {} // Wait until transmitter buffer is empty
c = EUSCI_A0 -> RXBUF; // Receive a character

if (c == 1)//UART mode
{
 for (i = 0; i < strlen(uart); i++)
      {
         while (! (EUSCI_A0->IFG & 0x02)) {}


 // Wait until transmitter buffer is empty
         EUSCI_A0->TXBUF = uart[i]; // Send a character
      }
```

```
                P2->OUT |= 1; // Turn ON P2.0 RED LED
  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 |= 0x80;
    }
    else if (c == 'B' || c == 'b')
    {
    P2->OUT |= 0x40;
    }
       else if (c == 'C' || c ==       'c')
    {
    P2->OUT |= 0x20;
    }
    else if (c == 'D' || c == 'd')
    {
    P2->OUT |= 0x10;
 }
                        else if (c == 'E' || c == 'e')
    {
    P4->OUT |= 0x01;
    }
```

```
    else if (c == 'F' || c == 'f')
    {
    P4->OUT |= 0x02;
    }
    else if (c == 'G' || c == 'g')
    {
    P4->OUT |= 0x04;
    }
    else if (c == 'H' || c == 'h')
    {
    P4->OUT |= 0x08;
}
else if (c == 'M' || c ==    'm')
{
break;
}
                                    else if (c== 'I' || c == 'i') //off


    {
                                        P2->OUT &= ~0x80;
P2->OUT &= ~0x40;
                                         P2->OUT &= ~0x20;
```

```
 P2->OUT &= ~0x10;
 P4->OUT &= ~0x01;
 P4->OUT &= ~0x02;
 P4->OUT &= ~0x04;
 P4->OUT &= ~0x08;

 }

 }
 break;
}
else if (c == 2)//button mode
{
 for (i = 0; i < strlen(button); i++)
     {
        while (! (EUSCI_A0->IFG & 0x02)) {}                    // Wait until transmitter
buffer is empty


        EUSCI_A0->TXBUF = button[i]; // Send a character
```

```
    }



P2->OUT |= 2; // Turn ON P2.1 Green LED
while(1) {
 while (! (EUSCI_A0-> IFG & 0x01)) {} // Wait until transmitter buffer is empty
  c = EUSCI_A0 -> RXBUF;              // Receive a character
    if (! (P1->IN & 0x40))
    {
    P2->OUT |= 0x80;
    }
    else if(! (P1->IN & 0x80))
    {
    P2->OUT |= 0x40;
    }
       else if(! (P3->IN & 0x01))
    {
    P2->OUT |= 0x20;



    }
    else if (! (P3->IN & 0x02))
```

```
    {
    P2->OUT |= 0x10;




    }
else if (! (P3->IN & 0x04))
    {

                        P4->OUT |= 0x01;
    }
                        else if (! (P3->IN & 0x08))
    {
            P4->OUT |= 0x02;
    }
    else if (! (P3->IN & 0x10))
    {
    P4->OUT |= 0x04;



    }
                        else if (! (P3->IN & 0x20))
    {
    P4->OUT |= 0x08;
}
```

```
else if (c == 'm' || c == 'M')
{
break;
                                }
else //off
   {
                                                P2->OUT &= ~0x80;
P2->OUT &= ~0x40;
P2->OUT &= ~0x20;
                                                 P2->OUT &= ~0x10;
                                               P4->OUT &= ~0x01;
                                                P4->OUT &= ~0x02;
P4->OUT &= ~0x04;
P4->OUT &= ~0x08;
}

} break;}
else if (c == 3)//automatic mode


{
 for (i = 0; i < strlen(automatic); i++)
```

```
    {
      while (! (EUSCI_A0->IFG & 0x02)) {}                    // Wait until transmitter
buffer is empty
      EUSCI_A0->TXBUF = automatic[i]; // Send a character
    }
```

```
P2->OUT |= 4; // Turn OFF P2.2 BLUE LED
while(1) {
while (! (EUSCI_A0-> IFG & 0x01)) {} // Wait until transmitter buffer is empty
```

```
   c = EUSCI_A0 -> RXBUF;
if(c=='m'||c=='M') {
break;
}
else{
```

```
                                        P2->OUT |= 0x80;
 delayMs(5000);
 P2->OUT &= ~0x80;
```

```
delayMs(5000);


P2->OUT |= 0x20;
                                        delayMs(5000);
P2->OUT &= ~0x20;
delayMs(5000);
P4->OUT |= 0x01;


delayMs(5000);
P4->OUT &= ~0x01;
delayMs(5000);


                                        P4->OUT |= 0x04;
delayMs (5000);
P4->OUT &= ~0x04;
delayMs (5000);
P2->OUT |= 0x40;
delayMs(5000);
P2->OUT &= ~0x40;
delayMs(5000);
P2->OUT |= 0x10;
delayMs(5000);
```

```
P2->OUT &= ~0x10;
delayMs(5000);
                                        P4->OUT |= 0x02;
delayMs(5000);
P4->OUT &= ~0x02;
delayMs(5000);



P4->OUT |= 0x08;
delayMs(5000);
                                         P4->OUT &= ~0x08;
delayMs(5000);

}
                }  break;
}
}
}
//PIN CONFIGURATION
void ipconfig(void) {

P2->SEL0 &= ~1;
P2->SEL1 &= ~1;
```

```
P2->DIR |= 1;
P2->SEL0 &= ~2;
P2->SEL1 &= ~2;
P2->DIR |= 2;
P2->SEL0 &= ~4;
P2->SEL1 &= ~4;
P2->DIR |= 4;
P2->SEL0 &= ~0x80;


P2->SEL1 &= ~0x80;
P2->DIR |= 0x80;
P2->SEL0 &= ~0x40;
P2->SEL1 &= ~0x40;
P2->DIR |= 0x40;
P2->SEL0 &= ~0x20;
P2->SEL1 &= ~0x20;
P2->DIR |= 0x20;
P2->SEL0 &= ~0x10;
P2->SEL1 &= ~0x10;
P2->DIR |= 0x10;


P4->SEL0 &= ~0x01;
P4->SEL1 &= ~0x01;
```

```
P4->DIR |= 0x01;



P4->SEL0 &= ~0x02;
P4->SEL1 &= ~0x02;
P4->DIR |= 0x02;



P4->SEL0 &= ~0x04;
P4->SEL1 &= ~0x04;
P4->DIR |= 0x04;
 P4->SEL0 &= ~0x08;


P4->SEL1 &= ~0x08;
P4->DIR |= 0x08;

P1->SEL1 &= ~0x40;
P1->SEL0 &= ~0x40;
P1->DIR &= ~0x40;
P1->REN |= 0x40;
P1->OUT |= 0x40;
```

```
P1->SEL1 &= ~0x80;


P1->SEL0 &= ~0x80;

P1->DIR &= ~0x80;
P1->REN |= 0x80;
P1->OUT |= 0x80;


P3->SEL1 &= ~0x01;
P3->SEL0 &= ~0x01;
P3->DIR &= ~0x01;
P3->REN |= 0x01;
P3->OUT |= 0x01;


P3->SEL1 &= ~0x02;
P3->SEL0 &= ~0x02;
P3->DIR &= ~0x02;
P3->REN |= 0x02;
P3->OUT |= 0x02;

P3->SEL1 &= ~0x04;
P3->SEL0 &= ~0x04;
P3->DIR &= ~0x04;
```

```
P3->REN |= 0x04;
P3->OUT |= 0x04;

P3->SEL1 &= ~0x08;
P3->SEL0 &= ~0x08;
P3->DIR &= ~0x08;
P3->REN |= 0x80;
P3->OUT |= 0x08;


P3->SEL1 &= ~0x10;
P3->SEL0 &= ~0x10;
P3->DIR &= ~0x10;
P3->REN |= 0x10;
P3->OUT |= 0x10;


P3->SEL1 &= ~0x20;
P3->SEL0 &= ~0x20;
P3->DIR &= ~0x20;
P3->REN |= 0x20;
P3->OUT |= 0x20;

}
// 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
                                // 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
}



 // Delay milliseconds when system clock is at 3 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 1 MS
}
```

IMPLEMENTATION STATUS:

The code is running without any errors and warnings.

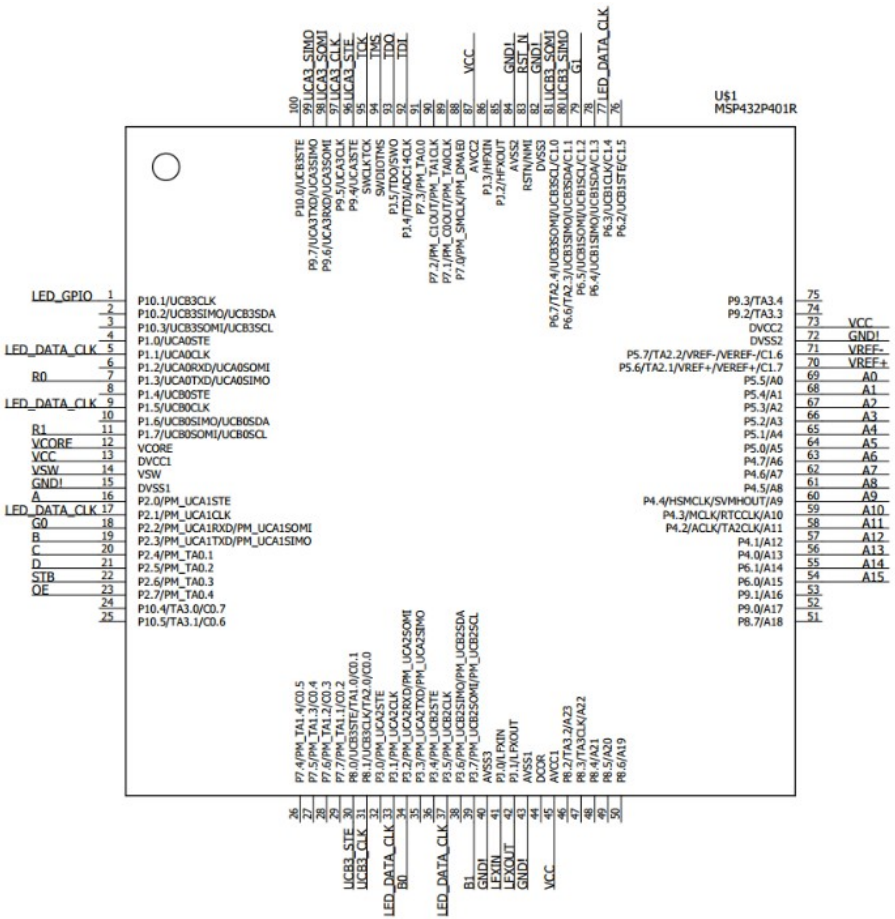
INFERENCE:

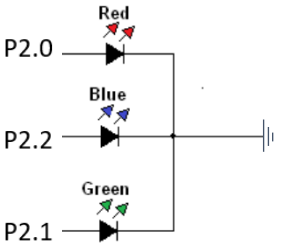
The code for designing a Robotic arm using the MSP432 microcontroller and UART communication involves initializing the necessary peripherals. The code enters an infinite loop and waits for user input from the terminal. Once a character is received, it checks the value of the character. If the character is '1', it enters the UART mode. It sends a message confirming the mode through UART and starts to glow RED LED and a nested loop to receive characters continuously. Based on the received characters, it sets or clears certain pins to control the MOTOR connected to the microcontroller's GPIOs.If the character is '2', it enters the SWITCH mode. It sends a message confirming the mode through UART and starts to glow GREEN LED and a nested loop to continuously check the state of specific buttons connected to the microcontroller's GPIOs. Based on the button states, it sets or clears certain pins to control the MOTORS. If the character is '3', it enters the automatic mode. It sends a message confirming the mode through UART and starts to glow BLUE LED and a nested loop to repeatedly toggle the state of certain pins to control the MOTORS with a predefined sequence.

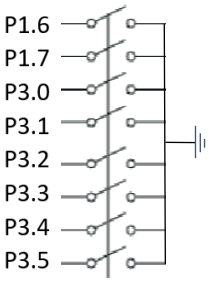
If the character is 'm' or 'M', it breaks the current mode loop and goes back to the main loop to wait for new input. Within each mode, if the character is not recognized or is a different control character, it turns off all the LEDs. Turning off all LEDs indicates that none of the loop started to run, it waits for the input to receive from UART.

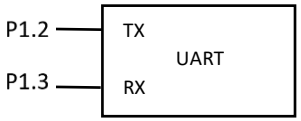
CONCLUSION:

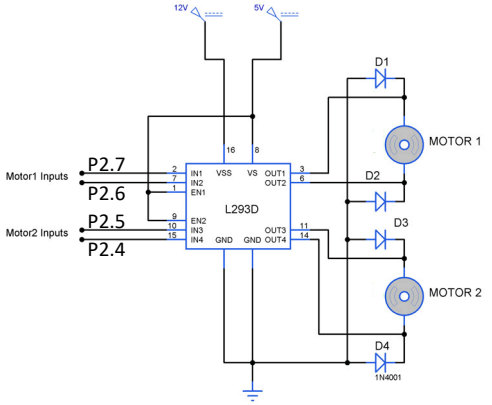
The provided code is designed to control a motor using the MSP432 microcontroller. It incorporates UART communication and button input to enable different modes of operation, including automatic, UART, and switch mode. The code utilizes pin configurations to control the motor's movement by sending appropriate signals to the motor driver.

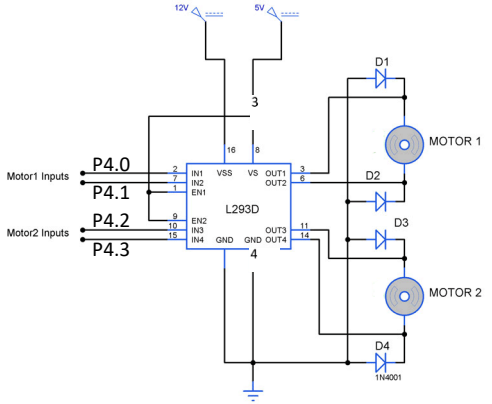

U$1
MSP432P401R
LED_GPIO
LED_DATA_CLK
R0
R1
VCORE
VCC
VSW
GND!
A
G0
B
C
D
STB
OE
P10.1/UCB3CLK
P10.2/UCB3SIMO/UCB3SDA
P10.3/UCB3SOMI/UCB3SCL
P1.0/UCA0STE
P1.1/UCA0CLK
P1.2/UCA0RXD/UCA0SOMI
P1.3/UCA0TXD/UCA0SIMO
P1.4/UCB0STE
P1.5/UCB0CLK
P1.6/UCB0SIMO/UCB0SDA
P1.7/UCB0SOMI/UCB0SCL
VCORE
DVCC1
VSW
DVSS1
P2.0/PM_UCA1STE
P2.1/PM_UCA1CLK
P2.2/PM_UCA1RXD/PM_UCA1SOMI
P2.3/PM_UCA1TXD/PM_UCA1SIMO
P2.4/PM_TA0.1
P2.5/PM_TA0.2
P2.6/PM_TA0.3
P2.7/PM_TA0.4
P10.4/TA3.0/C0.7
P10.5/TA3.1/C0.6
P9.3/TA3.4
P9.2/TA3.3
DVCC2
DVSS2
P5.7/TA2.2/VREF-/VEREF-/C1.6
P5.6/TA2.1/VREF+/VEREF+/C1.7
P5.5/A0
P5.4/A1
P5.3/A2
P5.2/A3
P5.1/A4
P5.0/A5
P4.7/A6
P4.6/A7
P4.5/A8
P4.4/HSMCLK/SVMHOUT/A9
P4.3/MCLK/RTCCLK/A10
P4.2/ACLK/TA2CLK/A11
P4.1/A12
P4.0/A13
P6.1/A14
P6.0/A15
P9.1/A16
P9.0/A17
P8.7/A18
VCC
GND!
VREF-
VREF+
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
LED_DATA_CLK
UCB3_STE
UCB3_CLK
LED_DATA_CLK
B0
LED_DATA_CLK
B1
GND!
LFXIN
LFXOUT
GND!
VCC


Red
P2.0
Blue
P2.2
Green
P2.1


P1.6
P1.7
P3.0
P3.1
P3.2
P3.3
P3.4
P3.5


P1.2
TX
UART
P1.3
RX


12V
5V
D1
MOTOR 1
Motor1 Inputs
P2.7
P2.6
L293D
D2
D3
Motor2 Inputs
P2.5
P2.4
MOTOR 2
D4
1N4001


12V
5V
D1
MOTOR 1
Motor1 Inputs
P4.0
P4.1
L293D
D2
D3
Motor2 Inputs
P4.2
P4.3
MOTOR 2
D4
1N4001