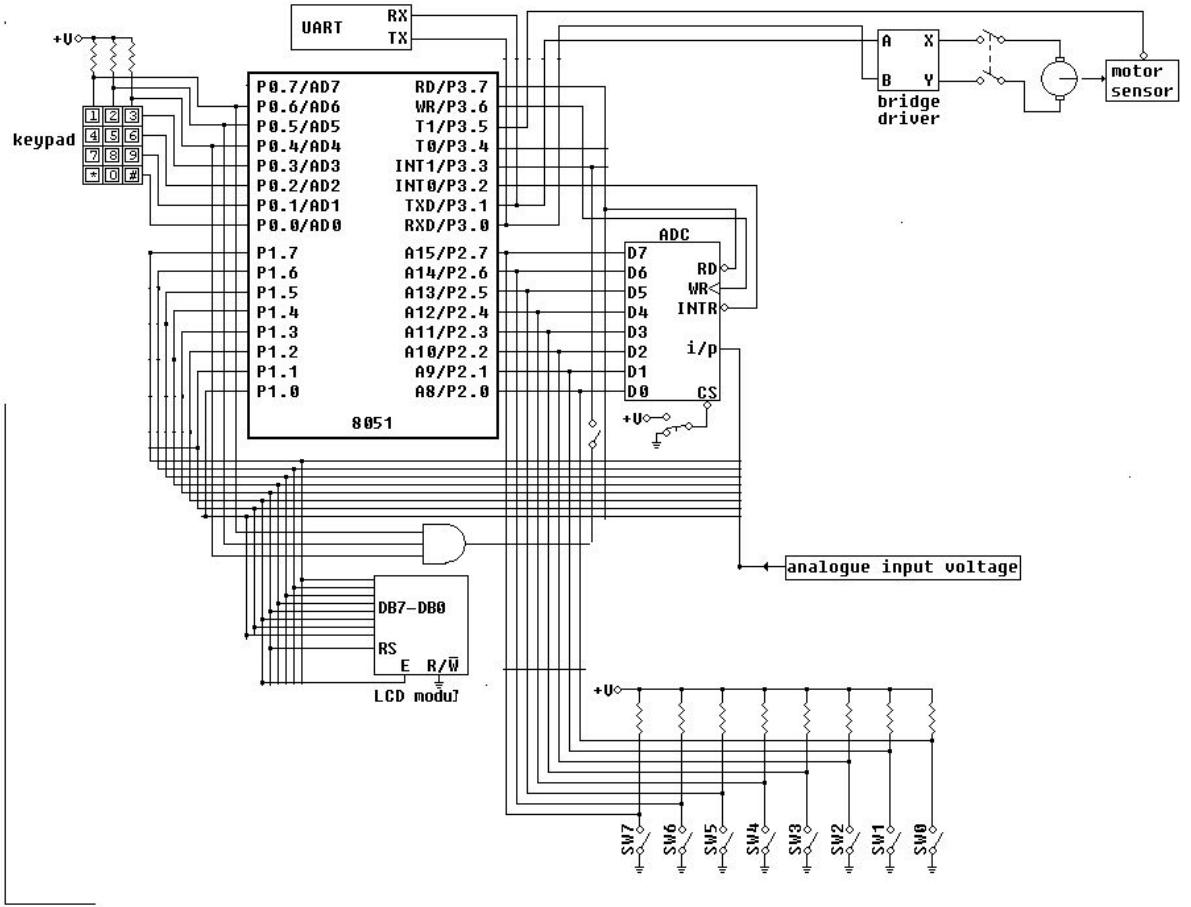
The Final Project Design

Batuhan Tosyalı and Enver Yiğitler

General Project Description

- The aim of this project is to create software to control the electric motor with peripheral devices such as serial port, keypad, ADC. The main function of the software is to set the speed of the motor with the keypad (gear) and ADC or values received from the serial port by the user. The user should lift the control switch to override the speed and gear values sending messages with the serial port.

Block Diagram



Hardware List

1. Intel 8051 Microcontroller
2. UART port
3. LCD display
4. Keypad
5. Switch
6. Timers
7. Bi-directional motor
8. Motor sensor

Task List

1. Motor task + Timer ISR + External ISR
2. Serial task + Serial ISR
3. Keypad task
4. LCD task + Timer ISR
5. Main task

Objectives of Tasks

- Motor task: Control the speed.
- Serial task: Communication and override protocol
- LCD and keypad task: Hardware abstraction
- Main task: Execute other tasks in a RR fashion
- No need for switch and ADC task

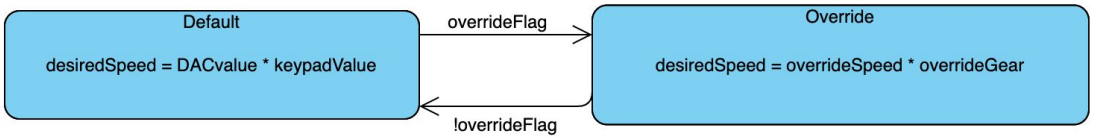
Motor Task + Timer ISR + External ISR

Variables: curSpeed, curAcceleration, desiredSpeed, revolution. iterationCount
Flags: overrideFlag

- External ISR will count the revolutions. It will be triggered by the motor sensor.
- Two jobs for Timer ISR
  - Signal the motor task periodically
  - Start and stop the motor according to desiredSpeed value. The range of the desiredSpeed will be 0-100 and it is used to determine the ratio of the period. The motor will be active during this duration and will be stopped for the rest of the period.
- The motor is started and stopped at the timer ISR to control the speed precisely.

Motor Task

- desiredSpeed will be calculated according to the overrideFlag. overrideFlag is controlled by the serial task



- The ADCvalue * keypadValue will be scaled by 100/2295 (255*9) to make the desiredSpeed between 0-100.
- curSpeed is calculated by dividing the revolution by period. curAcceleration is calculated by curSpeed - prevSpeed / period.

Pseudocode

```
timerISR()
            if(iterationCount < desiredSpeed)
                  start motor
            else
                  stop motor

            if(iterationCount >= 100)
                  iterationCount =0;

            motorTaskCounter++
            if(motorTaskCounter >= 100)
                  motorTaskFlag = 1;
                  motorTaskCounter = 0; //task period = timer period*100

externalISR() //triggered by motor sensor
            revolution++;
```

```
motorTask()
    if(motorTaskFlag)
        motorTaskFlag = 0;

        //read ACD value
        ADCvalue = readADC()

        //set desiredSpeed
        disableIRQ()
        if(overrideFlag)
            desiredSpeed = ADCValue * keypadValue * 100 /2295
        else
            desiredSpeed = overrideSpeed * overrideGear * 100 /2295
        enableIRQ()

        //calculate curSpeed and curAcceleration
        disableIRQ()
        curSpeed = revolution/period
        revolution = 0
        enableIRQ()
        curAcceleration = curSpeed - prevSpeed / period
        prevSpeed = curSpeed
```

Serial Task + Serial ISR

Variables: overrideSpeed, overrideGear, receiveBuffer
Flags: overrideFlag, msgReceived, acceptMessage

- Message format is: O(verride) + 3 digit speed + 1 digit gear +3 digit parity value+\n
  or STOPO(verride) + \n
- Parity value is calculated as the sum of all characters % 255
- Ex: O2555033\n // overrideSpeed = 255 , overrideGear = 5, parity value = 33
  STOPO\n

Serial Task

- This task will communicate with the serial port. It will respond to users with ACK/NAK depending on whether the message is valid and the switch is down

- If the message is valid and the switch is up, overrideSpeed and overrideGear will be assigned to values in the message and overrideFlag will be set.

- If the switch is turned off acceptMessage will be cleared and overrideFlag will be reset and the motor will pass to the default state. The switch will be polled in the main loop.

Pseudocode

```
serialISR()
      //set the flag if character send
      if(TI)
            msgSend = 1
            TI = 0

      if(RI)
            !! put the char into receiveBuf
            if( char = '\n')
                  msgReceived = 1
            RI = 0
```

```
serialTask()
        if(msgReceived)
            //clear flag
            msgReceived = 0
            disableIRQ()
            !! copy receiveBuf into msg
            enableIRQ()

            if(acceptMessage) //set by switch
            {
                if(msg is override operation)
                    if(checkParity(msg))
                        print("ACK\n")
                        overrideSpeed = getSpeed(msg)
                        overrideGear = getGear(msg)
                        overrideFlag = 1
                    else
                        print("NAK\n")
                else if(msg is stop command)
                    overrideFlag = 0
                    print("NAK\n")
            }
            else
                print("NAK\n")
```

Keypad Task

- Interaction with the keypad will be handled here. This task will search the key value and store it in a global variable (gearVal). The keypad will be polled in the main loop.

Pseudocode:

```
while(1)// in main loop
      gearVal = searchKey();
```

LCD Task + Timer ISR

- This task will display the measured speed, the measured acceleration. the desired speed, the motor state in the LCD display. Timer ISR is shared with the motor task.

Pseudocode:

```
timerISR()
      …
      drawToLCD = 1;

LCDTask()
      if(drawToLCD)
            drawToLCD = 0
            draw(desiredSpeed, curSpeed, curAccelaration, overrideFlag)
```

Thanks for listening

QA