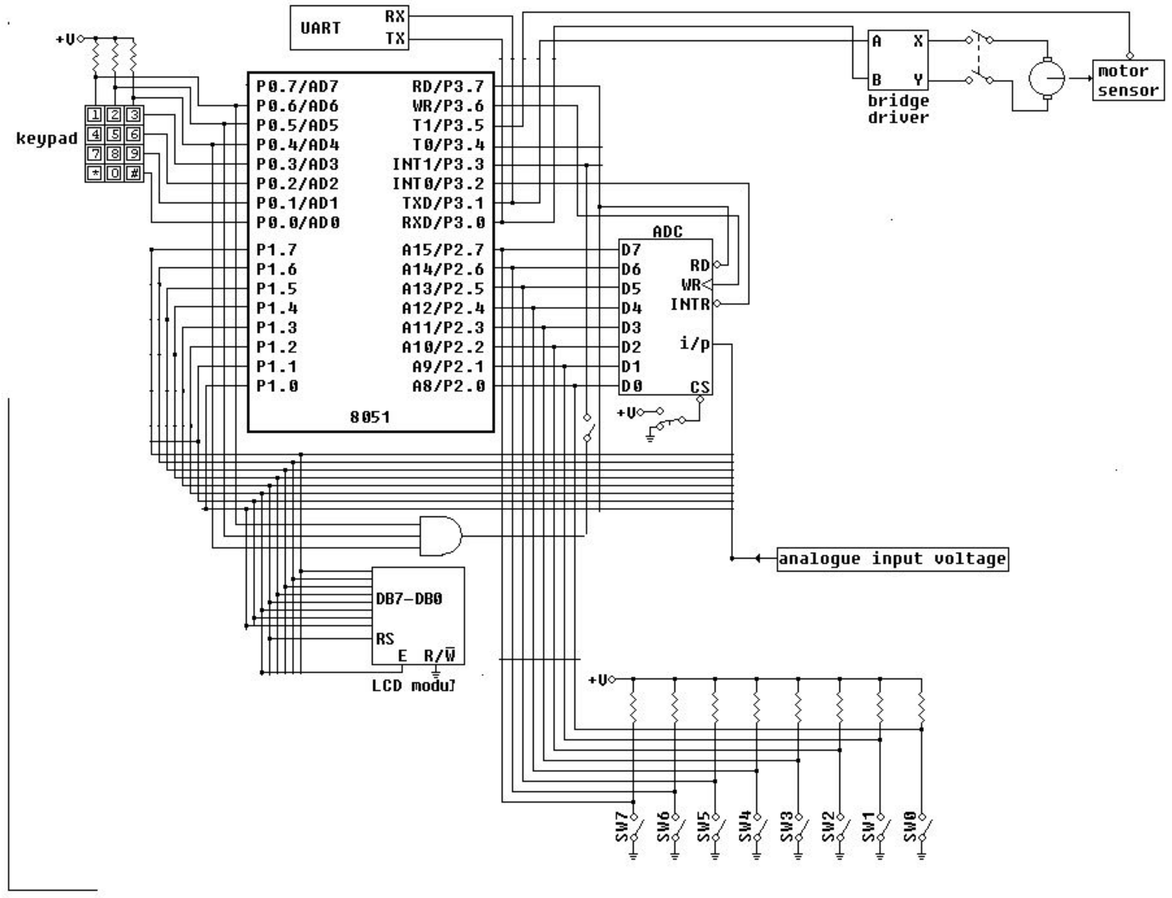
CS 431 - Embedded Systems
The Final Project Report

Enver Yiğitler 21702285
Batuhan Tosyalı 21702055

Block Diagram



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. The message protocol is defined below. We use edsim51 as our platform. We chose RR w/ISR as our software architecture.

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

General Design

The objective of the motor task is to adjust the speed of the motor. However, the speed of the motor cannot be controlled directly by the software. So we need to start and stop the motor by a ratio of a specified period. LCD and keypad tasks are for interaction with hardware. The serial task will handle communication and message protocol. The main task will only execute other tasks in a round-robin fashion. Motor task and LCD task will be signaled by the timer periodically. Timer ISR will be used by both LCD and motor tasks because there are only two timers in 8051 and timer1 is used by the serial task.
To count the number of rotations of the motor we will use the external ISR. External ISR will be triggered by the motor sensor. We choose not to create ADC and switch tasks because the required operation is simply reading the value and they don’t need hardware abstraction.

1. Motor Task + Timer ISR + External ISR

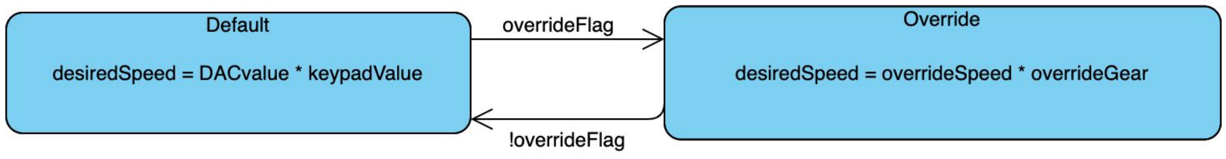
Variables: curSpeed, curAcceleration, desiredSpeed, revolution. iterationCount
Flags: overrideFlag

- External ISR will count the revolutions. It will be triggered by the motor sensor.

- Timer 1 ISR will be used by the motor task for two jobs with different periods. The first job is to signal the motor task periodically. The second job is to 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.

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


Default
desiredSpeed = DACvalue * keypadValue
overrideFlag
!overrideFlag
Override
desiredSpeed = overrideSpeed * overrideGear

- The DACvalue * 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 = 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
```

2. 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
  STOPO\n

- 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")
```

3. Keypad Task

Variables: gearVal

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

Pseudocode:

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

4. 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)
```