

MARKING CRITERIA AND STUDENT FEEDBACK

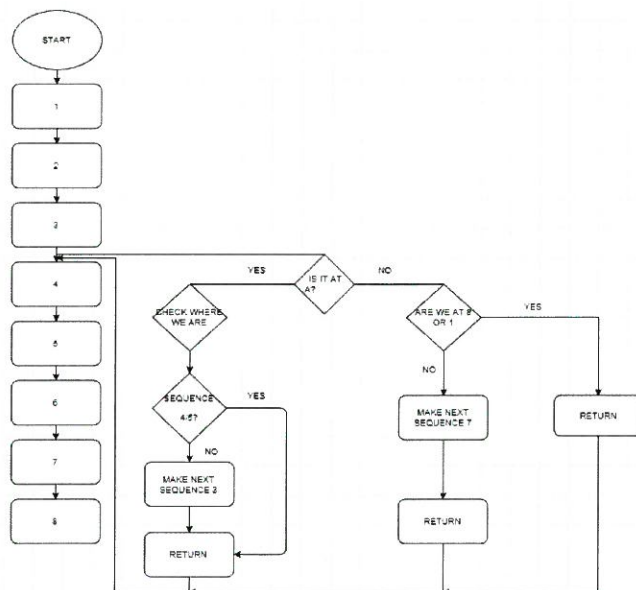
This section details the assessment criteria. The *extent* to which these are demonstrated by you determines your mark. The marks available for each criterion are shown. Lecturers will use the space provided to comment on the achievement of the task(s), including those areas in which you have performed well and areas that would benefit from development/improvement.

Common Assessment Criteria Applied	Marks available	Marks awarded
1. Research-informed Literature Extent of research and/or own reading; selection of credible sources; application of appropriate referencing conventions.		
Very little evidence of reading. Poor referencing.	10	2
2. Knowledge and Understanding of Subject Extent of knowledge and understanding of concepts and underlying principles associated with the discipline.		
Some knowledge indicated - a program was generated which did assemble without errors. Program did not meet criteria however	30	10
3. Analysis Analysis, evaluation and synthesis; logic, argument and judgement; analytical reflection; organisation of ideas and evidence		
Some analysis of Program deficiency shown.	10	02
4. Practical Application and Deployment Deployment of methods, materials, tools and techniques; application of concepts; formulation of innovative and creative solutions to solve problems.		
The program contained some good points esp especially the control of timers. However the documentation was very poor as was the commenting within the program	45	12
5. Skills for Professional Practice Attributes in professional practice: individual and collaborative working; deployment of appropriate media; presentation and organisation.		
Overall quality of report is not at level required.	5	0
Assignment Mark Overall (Assessment marks are subject to ratification at the Exam Board. These comments and marks are to give feedback on module work and are for guidance only until they are confirmed.)		
Late Submission Penalties (tick if appropriate) Up to one-week late ☐ Over one week late ☐		25. %

TASK 1 A)**Introduction**

The task I have been set is to provide traffic flow control at the intersection of two roads, using a traffic light system. The sequence has been provided for me. I have added an interrupt and various subroutines to make it a fully operational system. The microcontroller I am going to use to control this system is a PIC16F84A. After the sequence has run all the way through, it returns to the start and it runs through again in a loop. Writing the programme itself on the MPLABX software, where I can test the programme, edit and annotate it.

I have designed a flow chart to demonstrate how my program will work including the interrupt.

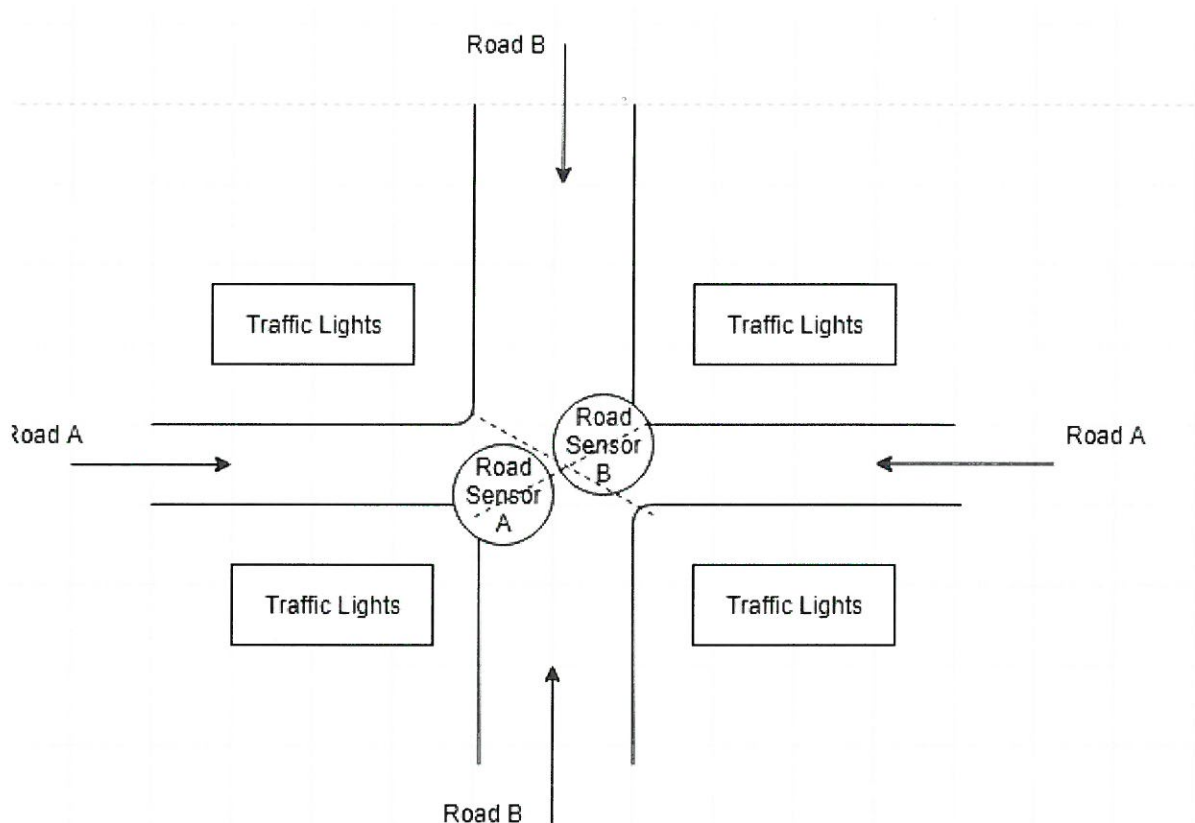


As you can see the interrupt has been triggered in-between 3&4. If a car arrives at road A or B, A sensor will set off the interrupt. First we check where the interrupt has been called from 'is it at a?' once we know where the car is the sequence jumps to sequence number 3. If it is at number 4/5 however it continues. If the car is at road B then the sequence will go to number 7. If it is at number 8/1 it continues as well. If the car is at road A no matter what the outcome is the program returns to where it left off. This is an example of temporary memory off the stack.

In my code, I have worked in time delays between each stage, as you can see each number in the sequence has a specific time:

sequence number	time seconds	road a			road b		
		red 1	amber 1	green 1	red 2	amber 2	green 2
1	40	1	0	0		0	1
2	5	1	0	0	0	1	0
3	5	1	1	0	1	0	0
4	1	0	1	0	1	0	0
5	40	0	0	1	1	0	0
6	5	0	1	0	1	0	0
7	5	1	0	0	1	1	0
8	1	1	0	0	0	1	0

This shows how there is only traffic flowing parallel and never perpendicular. It is a two way system. Where there is a '1' the traffic light is on. With traffic lights it is very important that everything is running on the same delays, so that you don't have one light changing out of sequence with the others.



This shows the basic layout of the system, including where my interrupt triggers are the road sensors

Equating

There are a few different routes I could of gone down but this is the method I decided on. Here I have equated the lights to their addresses. I am using hex.

```
portB equ 0x06 ;defines port address
```

```
count equ 0x0c
```

```
nCountequ    0x0d
mCount      equ    0x0e
red    equ    0x0f    ; red lamp on
redamb      equ    0x10    ;red amber on
green equ    0x11    ;green on
amber equ    0x12    ;amber on
countx equ    0x13    ; temp store of time
counts equ    0x14
```

(my full code will be attached in a word document)

Deficiencies

The main issue I can see with my program is to do with the interrupt, if two sensors are triggered at once. For example, the lights are green for road A for 5 seconds, as a car appears at road B, it would trigger the interrupt signal. Only allowing road B, 5 seconds for the green light to appear until the next cycle, which in turn could slow down the flow of traffic. The system doesn't consider that some roads might be busier than others at different times of day. The interrupt is the same in every case. Where this program works in this case, a more sophisticated one needs to be designed if I took this task any further.

2

All information has been gathered from PIC16F84a Reference manual and 'PIC programming in assembly'

(www.mstracey.btinternet.co.uk/index.htm)

Here is the full code:

processor 16f84a

include <p16f84a.inc>

__config _RC_OSC & _WDT_OFF & _PWRTE_ON

radix hex

portB equ 0x06 ;defines port address

count equ 0x0c

nCount equ 0x0d

mCount equ 0x0e

red equ 0x0f ; red lamp on

redambequ 0x10 ;red amber on

green equ 0x11 ;green on

amber equ 0x12 ;amber on

countx equ 0x13 ; temp store of time

counts equ 0x14

org 0x000

goto start

;

org 0x04

retfie

;

start movlw 0x00

tris portB

```
    clrf    portB

    movlw  0x0A

    movwf  red

    movlw  0x0A

    movwf  redamb

    movlw  0x0A

    movwf  green

    movlw  0x0A

    movwf  amber

redlite movlw  0x01

    movwf  portB

    movf   red

    movwf  countx

rloop  call   pause

    decfsz countx

    goto   rloop

;

    movlw  0x03

    movwf  portB

    movf   redamb

    movwf  countx

reloop call   pause

    decfsz countx

    goto   reloop

;

    movlw  0x04

    movwf  portB
```

```
    movf  green
    movwf countx
reelooptcall    pause
    decfsz countx
    goto  reeloopt
;
```

```
    movlw 0x02
    movwf portB
    movf  amber
    movwf countx
aloopt call    pause
    decfsz countx
    goto aloop
```

```
goto redlite
```

```
pause movlw 0xff
    movwf mCount

loadN movlw 0xff
    movwf nCount
decN  decfsz nCount, f
    goto decN
    decfsz mCount, f
    goto loadN

return
```

end

TASK 2

In this task I will be controlling a DC motor using pulse width modulation. Pulse width modulation is controlled using parameters like, load voltage and current. I will be using a dc square wave input to power my circuit. The electrical energy is converted into physical movement. Dc motors are the simplest common type of electrical motor and can be found everywhere. The direction of rotation of the motor is changed by changing the direction of current flow using a 'H bridge' configuration. PICS produce a square wave output of 0-5v. The motor requires a 12v source, which would fry the chip if directly connected. The method of interfacing I would use is a transistor connected to a relay, which has a diode across it to protect against induced current.