

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.		
There is no evidence of referencing or other reading.	10	0
2. Knowledge and Understanding of Subject Extent of knowledge and understanding of concepts and underlying principles associated with the discipline.		
you show a reasonable understanding and your program designs have some merit. Your approach to determining delays could have been quite good if fully implemented. You still have some more to learn regarding interrupts.	30	20
3. Analysis Analysis, evaluation and synthesis; logic, argument and judgement; analytical reflection; organisation of ideas and evidence		
your analysis is quite good.	10	5
4. Practical Application and Deployment Deployment of methods, materials, tools and techniques; application of concepts; formulation of innovative and creative solutions to solve problems.		
A fair attempt - you would gain marks with more narrative within and around your program. It is clear that the program did not fully meet the requirements but many of the elements did work.	45	24
5. Skills for Professional Practice Attributes in professional practice: individual and collaborative working; deployment of appropriate media; presentation and organisation.		
you would gain marks with S/W generated flow charts and a referencing section.	5	3

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

52.2%

LUKE HICKS

Microcontrollers

Assignment

Task 1

Part A)

To design and create a traffic control system that will be used on a busy intersection of two roads requires a lot of planning. This was done beforehand on paper before any programming had taken place. These hardcopies have been scanned into PDF for easier access and are attached with this document to show the order in which the programme will operate.

The traffic lights will follow a set sequence with 8 different lighting positions which are illustrated in the chart below. However this sequence can be interrupted if a sensor is pressed. The interrupt and delay are also illustrated via flow charts on separate PDF's attached to this document.

Equipment

- Breadboard
 - LED's (Red, Amber, Green)
 - 5v Power Supply
 - Resistor's
- 

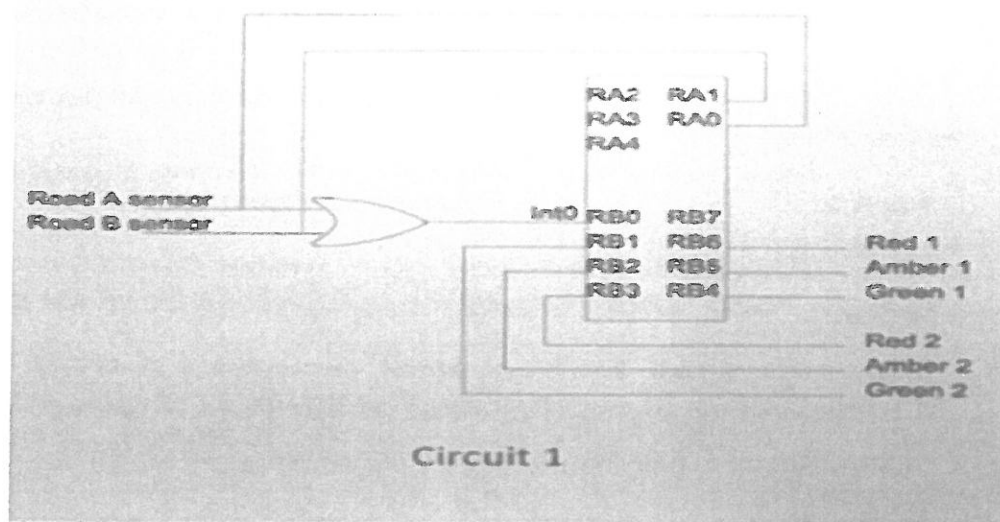
Software

- MPLAB X IDE
- Computer / Laptop

Sequence

Sequence No	Time seconds	Road A			Road B		
		Red 1	Amber 1	Green 1	Red 2	Amber 2	Green 2
1	40	1	0	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

Circuit



Code (MPLAB XIDE Screenshots)

```

13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106

;-----
; PortB equ 06
; PortA equ 05
; pause1 equ 0x0c
; pause2 equ 0x0d
; pause3 equ 0x0e
; intcon equ 0x0b
;-----

org 0x00
goto start
org 0x04

start
    movlw 0x01      ;port A is set to input & port B to output
    tris   PortB
    movlw 0xff
    tris   PortA
    BSF    intcon,7
    BSF    intcon,4
    BCF    intcon,1

loop
    movlw 0x42      ;42 to port B
    movwf PortB
    call pause
    call pause

    call pause
    call pause

    movlw 0x44      ;44 hex to port B
    movwf PortB
    call pause
    call pause

    movlw 0x68      ;68 hex to port B
    movwf PortB
    call pause
    call pause

    movlw 0x28      ;28 hex to port B
    movwf PortB
    call pause

    movlw 0x18      ;18 hex to port B
    movwf PortB
    call pause
    call pause
    call pause
    call pause
    call pause
    call pause

    movlw 0x28      ;28 hex to port B
    movwf PortB
    call pause

    movlw 0x4c      ;4c hex to port B
    movwf PortB
    call pause
    call pause

    movlw 0x44      ;44 hex to port B
    movwf PortB
    call pause
    goto loop

pause
    movlw 0xff
    movwf pause1
    movlw 0xff
    movwf pause2
    movlw 0x11
    movwf pause3

loop1
    decfsz pause1,1 ;decrementing by 1 from 255
    goto loop1
    decfsz pause2,1 ;decrementing by 1 from 255
    goto loop1
    decfsz pause3,1 ;decrementing by 1 from 17
    goto loop1
    RETURN
    END
;-----

```

*But Timer
Not fully implemented*

Code

```
; Author:   Luke Hicks           ;
; Date:     14/03/2018          ;
; Version:   1.2                 ;
; Title:     Traffic Lights      ;
; Description: Two Way Traffic Control ;
```

```
processor 16f84a
```

```
include <p16f84a.inc>
```

```
__config _RC_OSC & _WDT_OFF
```

```
radix hex
```

```
;-----;
```

```
PortB equ 06
```

```
PortA equ 05
```

```
pause1 equ 0x0c
```

```
pause2 equ 0x0d
```

```
pause3 equ 0x0e
```

```
intcon equ 0x0b
```

```
;-----;
```

```
org 0x00
```

```
goto start
```

```
org 0x04
```

```
start                                ;port A is set to input & port B to output ;
```

```
movlw 0x01
```

| tris | PortB |

```
movlw 0xff
```

| tris | PortA |
BSF intcon,⁷

BSF intcon,4

BCF intcon,1

```
loop          ;42 to port B          ;
```

```
movlw 0x42      ;42 is put into the w register. 1-red 2-green ;
```

movwf PortB

call pause

call pause

call pause

call pause

call pause

call pause

```

;44 hex to port B

```

```
movlw 0x44      ;44 is put into the w register. 1-red 2-amber ;
```

movwf PortB

call pause

call pause

```

                                ;68 hex to port B                                ;
movlw 0x68                    ;68 is put into the w register. 1-red,amber 2-red;
movwf PortB
call pause
call pause

                                ;28 hex to port B                                ;
movlw 0x28                    ;28 is put into the w register. 1-amber2 2-red ;
movwf PortB
call pause

                                ;8 hex to port B                                ;
movlw 0x18                    ;18 is put into the w register. 1-green 2-red ;
movwf PortB
call pause
call pause
call pause
call pause
call pause
call pause

                                ;28 hex to port B                                ;
movlw 0x28                    ;28 is put into the w register. 1-amber 2-red ;
movwf PortB
call pause
call pause

                                ;4c hex to port B                                ;
movlw 0x4c                    ;4c is put into the w register. 1-red 2-red,amber;
movwf PortB
call pause

```

```
call pause  
  
;output 44 hex to port B ;  
  
movlw 0x44 ;44 is put into the w register. 1-red 2-amber ;  
movwf PortB  
call pause  
goto loop
```

```
pause movlw 0xff  
movwf pause1  
movlw 0xff  
movwf pause2  
movlw 0x11  
movwf pause3
```

```
loopL decfsz pause1,1 ;decrementing by 1 from 255  
goto loopL  
decfsz pause2,1 ;decrementing by 1 from 255  
goto loopL  
decfsz pause3,1 ;decrementing by 1 from 17  
goto loopL  
RETURN  
END
```

Analysis

```

1  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
2  ;                               Author :    Luke Hicks                               ;
3  ;                               Date:      14/03/18                               ;
4  ;                               Version:   1.2                               ;
5  ;                               Title:     Traffic Lights                       ;
6  ;                               Description: Two way Traffic Control             ;
7  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
8  ;                               Note:      Tested on PIC16F84                     ;
9  ;                               List      F=16F84                               ;
10 ;                               Radix     Hex                                   ;
11 ;-----;
12

```

When writing any code it is important to always start with information about what the code is doing and what equipment it will be used on in this case a PIC16F84, along with name of the designer and date. The picture above shows the recommended amount of information to include before you start the code.

```

;-----;
    PortB equ 06
    PortA equ 05
    pause1 equ 0x0c
    pause2 equ 0x0d
    pause3 equ 0x0e
    intcon equ 0x0b
;-----;

    org 0x00
    goto start
    org 0x04

```

This is done before you start your code as the picture says it's a memory map. This is where you can assign addresses name so when you are writing you can call it but the name rather than the address which make things easier for both people writing the code or reading it. For example MyPortA is in address 5.

Part B)

An interrupt is added to the code to help the system run more efficiently, this is because if there is no traffic on one road but some on the other the interrupt will begin. This interrupt goes from the normal sequence to a part where it allows the lights to change to prevent traffic waiting for no need. This is done by a pressure sensor on the road which shows a car waiting and for the sequence to change to allow them to get going again. An interrupt can happen at any time so the sequence will just continue as normal before an interrupt and again after. When the sensor is pressed either A or B it closes one of the switches which allows there to be voltage at the port either RA0 or RA1. RB0 then reads the input ports to see which sensor has been pressed and it will know this because there will be voltage on the port. It will then start whichever sequence is need for example RA1 would mean senor A has been pressed and to initiate the light sequence for that road and vice versa for sensor B. **(Sequences are shown on flow charts attached as PDF).**

```

25
26
27      start                ;port A is set to input & port B to output
28          movlw    0x01
29          tris     PortB
30          movlw    0xff
31          tris     PortA
32          BSF      intercon,7
33          BSF      intercon,4
34          BCF      intercon,1
35

```

Intercon, 7 is universal interrupt which cause the programme to deviate from the main routine. Then intercon, 4 activates it on RBO to the correct sequence call then intercon, 1 allows the programme to return back to the normal routine.

Part C)

A deficiency that may exist in the algorithm is if both sensors were pressed which one would get priority. Also it is important that this doesn't overload the software and cause it to crash it would lead to inconvenience to road users. So this would have to be resolved maybe a use of a timer so the sensors work once a car has been working for a certain amount of time before the interrupt is caused. Also another problem would be if there was a pedestrian crossing also at the lights because this would mean a different interrupt could be called at any given time and both lights would have to be red. Again who would get priority over the interrupt and how long would the interrupt take to begin once the sensors have been pressed.

Another interrupt could be called as modern traffic systems change sequence for buses if they are running like the ones Swansea City Council have implemented across the city. This would also cause an issue with which interrupt has priority at busy times.

Also to improve the programme I have written could find another way to cause a time delay instead of using call pause numerous times. This is because it adds length to the programme and makes it harder to read, it is not very professional to have to call a time delay more than once. Once I continue to improve my programme that is the first issue to be addressed. ✓

Task 2

The task issued was to write a programme to control the speed of a motor module using Pulse Width Modulated (PWM) signal. The PWM control required a 5V that has duration of 0.5mS and 1.5mS this was repeated after 50mS. There are flow charts to help show the planning of the programme and what the programme does attached via PDF for easy access.

Equipment

- Motor Module
- 5v Power Supply
-

Software

- MPLAB X IDE
- Computer / Laptop
- BCD Display's x2

- I. 0.5mS results in the motor rotating full speed in an anticlockwise direction
- II. 1.0mS results in the motor being stopped
- III. 1.5mS results in the motor rotating full speed in a clockwise direction.

The speed will be increased / decreased in 0.01mS increments with there are 50 speeds setting available in both clockwise and anti-clockwise directions. Switch A will control the speed increments where switch B will be the direction in which the motor turns. Switch C is the overriding control which latches, this means that when the will either be on or off. On/Closed means that the motor is running and off/open means that the motor is on stop.

Setting Number	BCD 2	BCD 1	Setting Number	BCD 2	BCD 1
1	0000	0001	26	0001	1010
2	0000	0010	27	0001	1011
3	0000	0011	28	0001	1100
4	0000	0100	29	0001	1101
5	0000	0101	30	0001	1110
6	0000	0110	31	0001	1111
7	0000	0111	32	0010	0000
8	0000	1000	33	0010	0001
9	0000	1001	34	0010	0010
10	0000	1010	35	0010	0011
11	0000	1011	36	0010	0100
12	0000	1100	37	0010	0101
13	0000	1101	38	0010	0110
14	0000	1110	39	0010	0111
15	0000	1111	40	0010	1000
16	0001	0000	41	0010	1001
17	0001	0001	42	0010	1010
18	0001	0010	43	0010	1011
19	0001	0011	44	0010	1100
20	0001	0100	45	0010	1101
21	0001	0101	46	0010	1110
22	0001	0110	47	0010	1111
23	0001	0111	48	0011	0000
24	0001	1000	49	0011	0001
25	0001	1001	50	0011	0010

NOT
BCD

The chart above shows how the program controls the motor using an operator interface. The speed of the motor will be shown on two Binary Coded-Decimal (BCD) displays which are attached to the processor. Ports RB4-RB6 are used for BCD 1 & ports RB0-RB3 are used for BCD 2. Then port RB7 is used for the PWM input on the D.C. Motor control board.

Code

```

; Author: Luke Hicks
; Date: 11/04/2018
; Version: 1.2
; Title: D.C. Motor Control
;
; Description: D.C. Motor Control

processor 16f84a
include <p16f84a.inc>
__config _RC_OSC & _WDT_OFF & _PWRTE_ON

portB equ 0x06
portA equ 0x05
timer equ 0x0c
count equ 0x0d

org 0x000

goto set

```

```
set    movlw 0xff
      tris   portA

      movlw 0x00
      tris   portB

      movlw D'150'
      movwf count

intro  BTFSS portA,3
      goto   startb
      goto   start

start  BTFSC portA,1
      goto   start

      BTFSS portA,2
      INCFSZ    count
      goto   check
      DECF    count

check  BTFSS portA,0
      DECFSZ    count
      goto   route1
      INCF    count,d'100'

startb BTFSC portA,1
```

```
goto startb
```

```
BTFSS portA,2
```

```
INCFSZ count
```

```
goto check1
```

```
DECF count
```

```
check1 BTFSS portA,0
```

```
DECFSZ count
```

```
goto route1
```

```
INCF count,d'100'
```

```
; Output high and delay ;
```

```
route1 movlw 0x03
```

```
movwf portB
```

```
movf count,w
```

```
movwf timer
```

```
repl DECFSZ timer
```

```
goto repl
```

```
; Output low and delay ;
```

```
movlw 0x00
```

```
movwf portB
```

```
movf count,w
```

```
movwf timer
```

Luke Hicks

```
comf timer
```

```
INCF timer
```

```
rep2 DECFSZ timer
```

```
goto rep2
```

```
goto intro
```

```
; Output high and delay ;
```

```
route2 movlw 0x05
```

```
movwf portB
```

```
movf count,w
```

```
movwf timer
```

```
rep3 DECFSZ timer
```

```
goto rep3
```

```
; Output low and delay
```

```
movlw 0x00
```

```
movwf portB
```

```
movf count,w
```

```
movwf timer
```

```
comf timer
```

```
INCF timer
```

```
rep4 DECFSZ timer
```

Luke Hicks

goto rep4

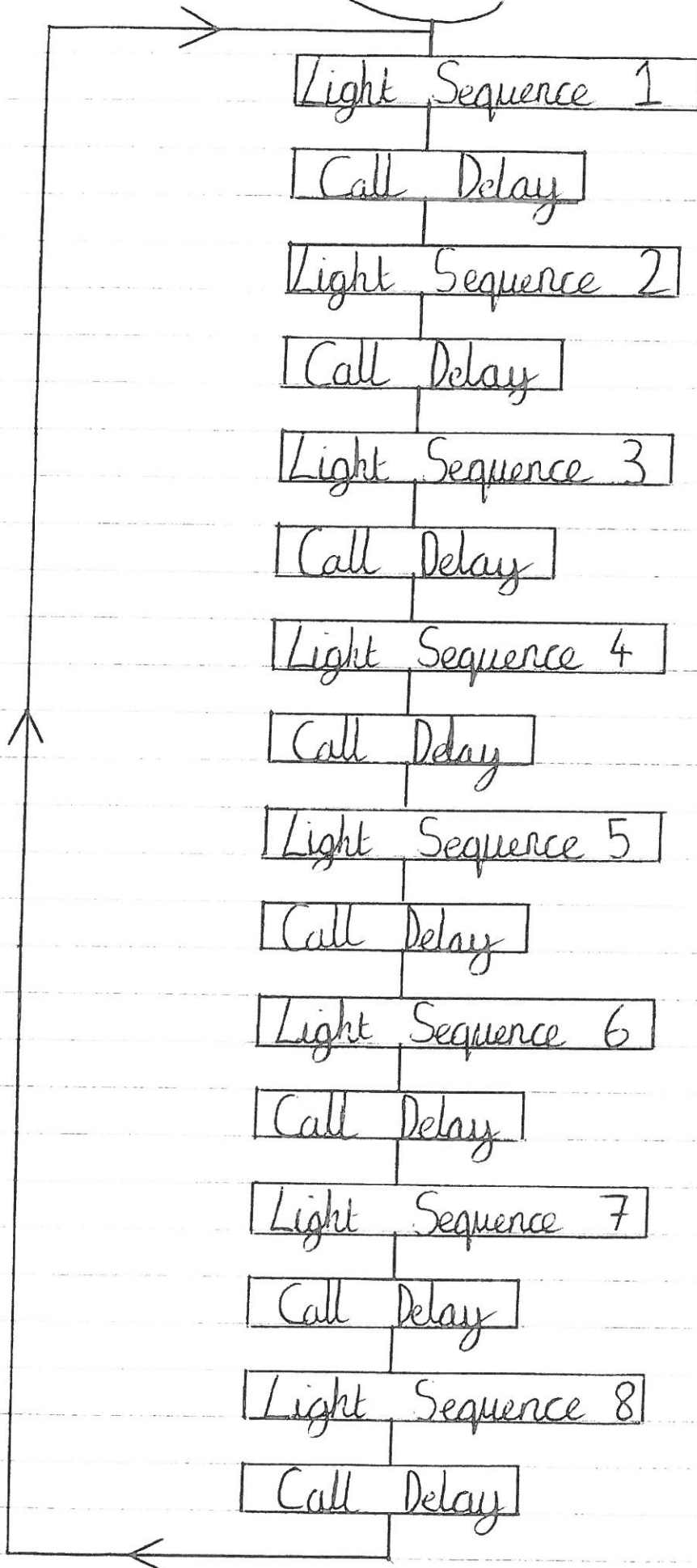
; Repeat

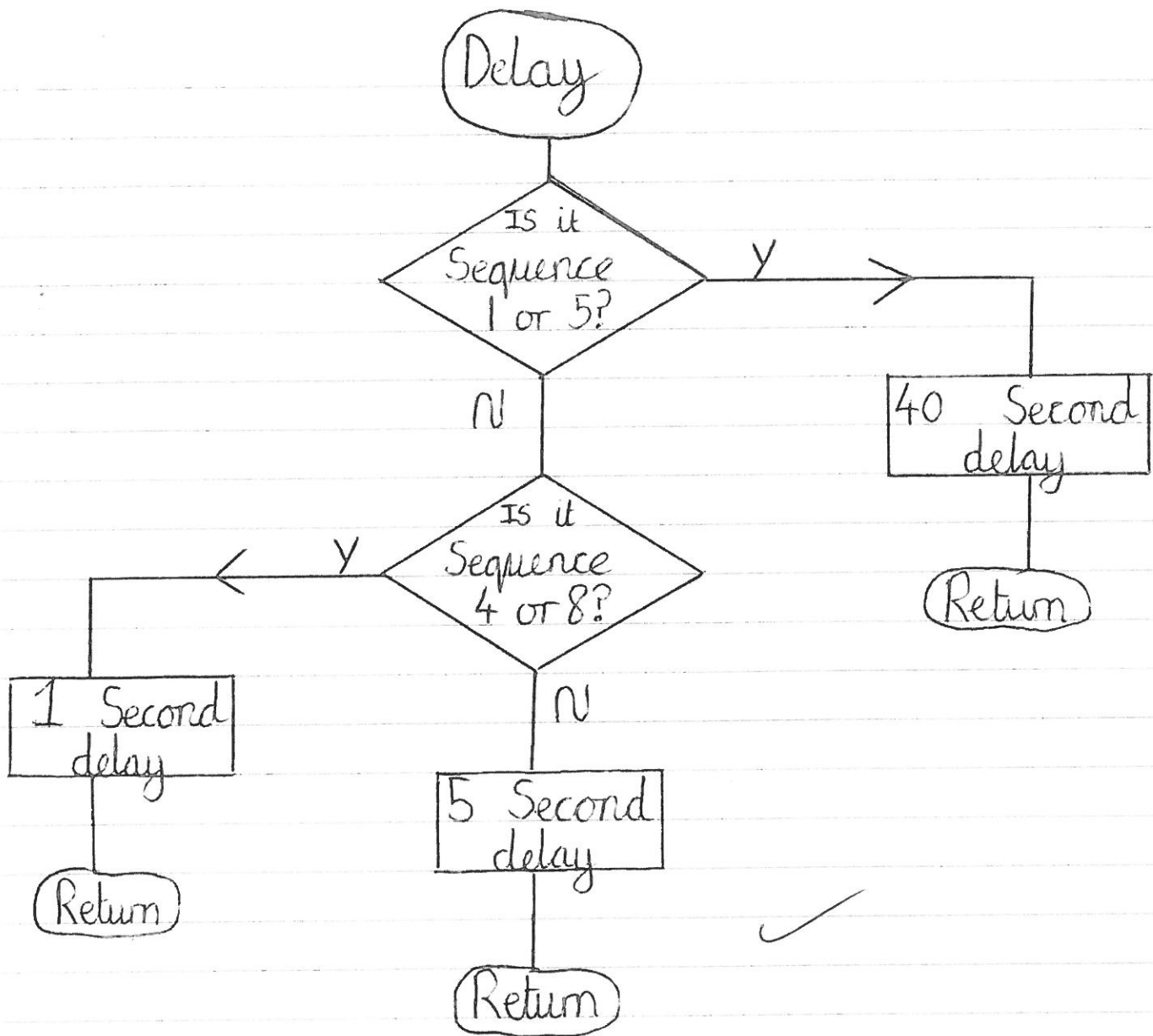
goto intro

end

Start

LUKE HICKS





Interrupt

LUKE HICKS

Read Port A

Sensor A or B Pressed?

Check Sequence

Is it Sequence 4 or 5?

Return

Jump to Sequence 3 +

Return

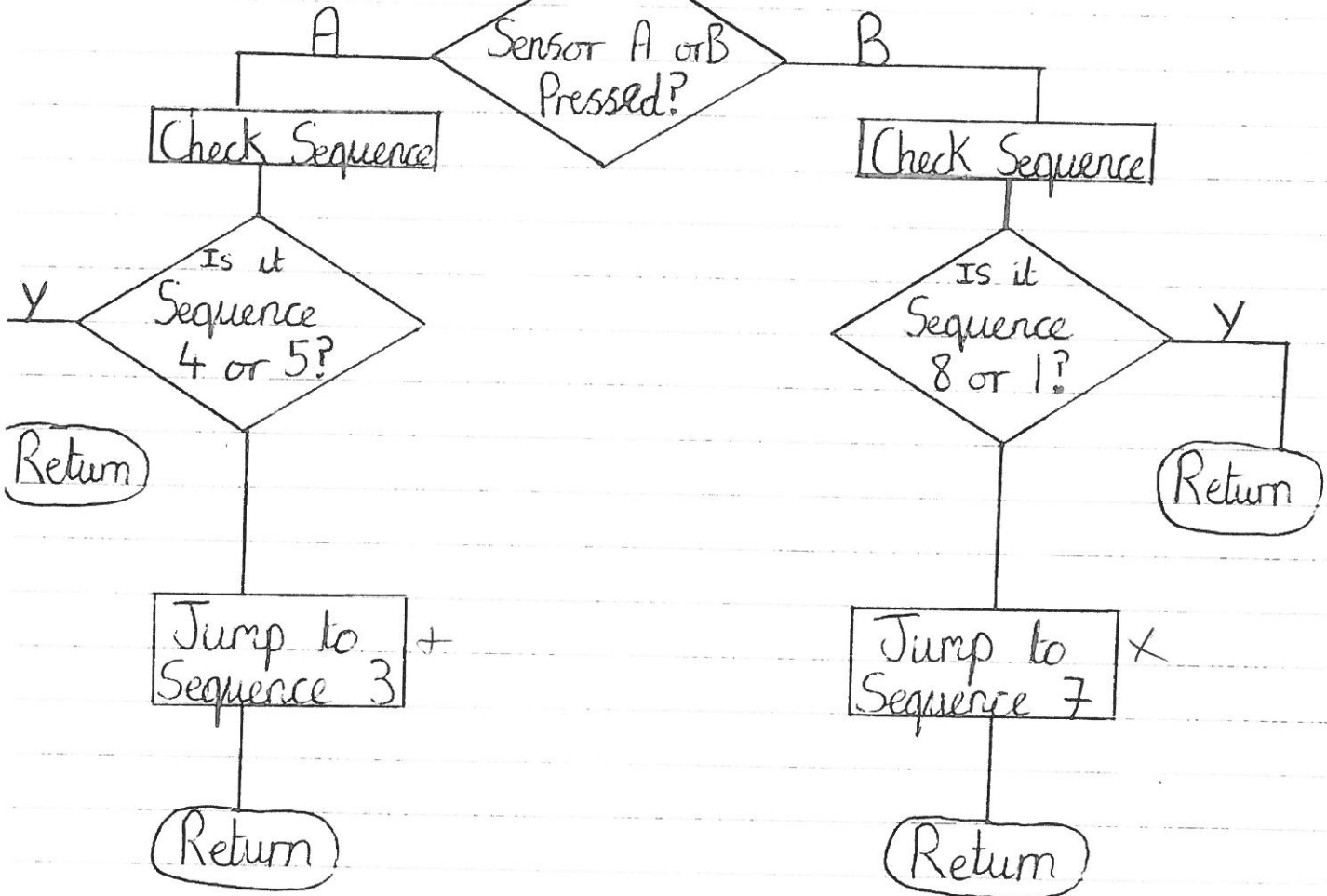
Check Sequence

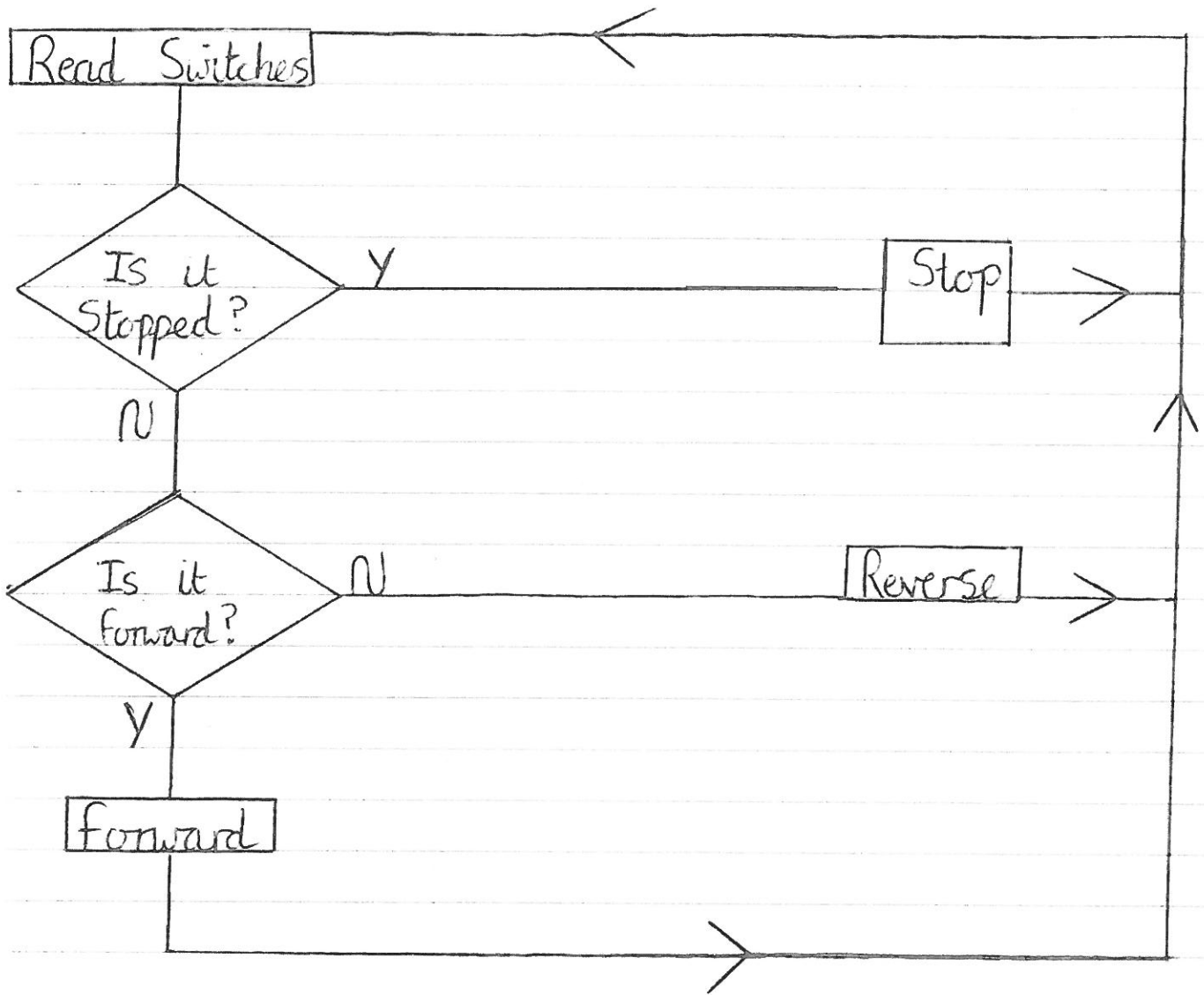
Is it Sequence 8 or 1?

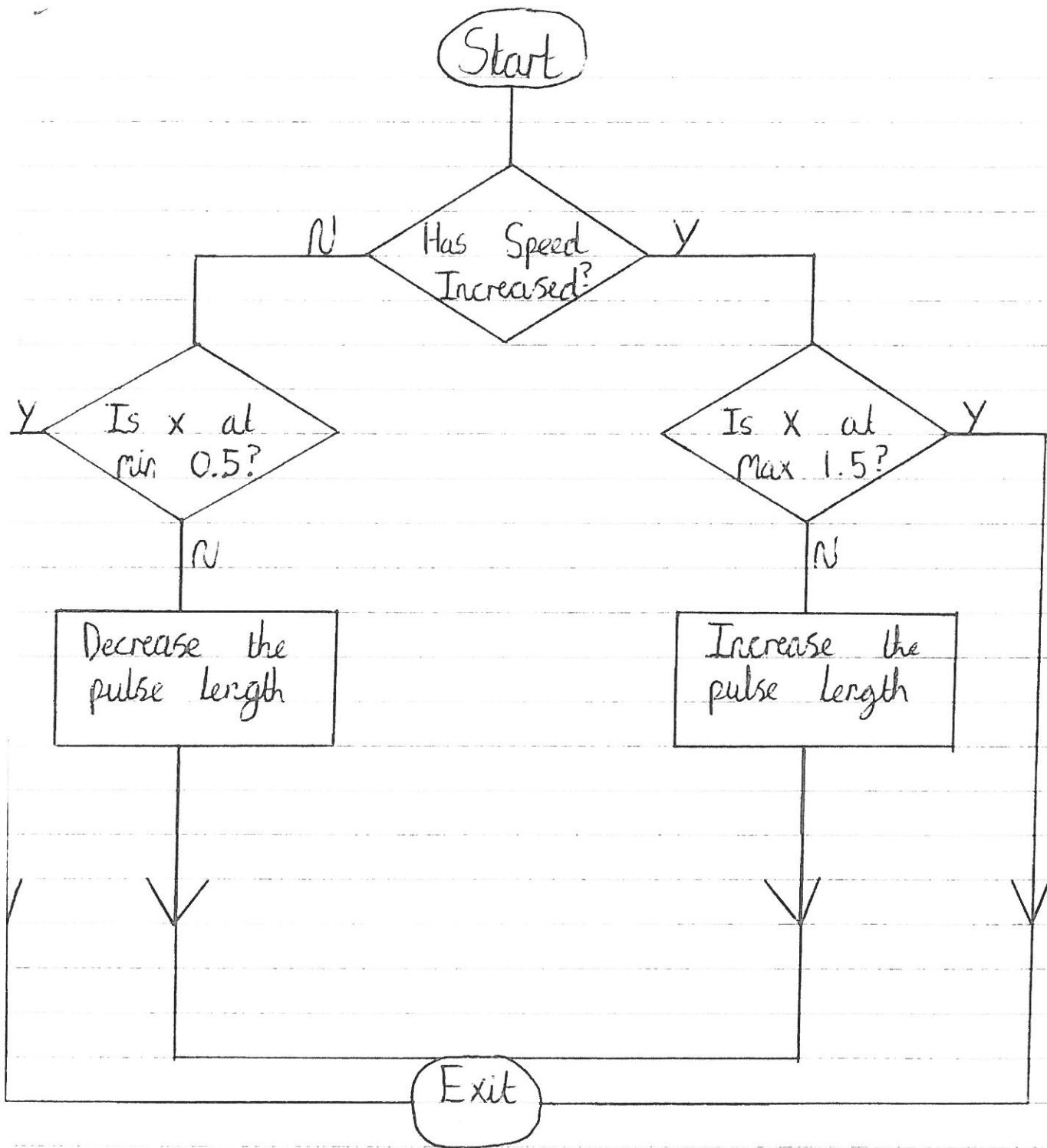
Return

Jump to Sequence 7 x

Return







Luke Hicks

