

Complete, fold and seal
(see note 2 below)

SURNAME: TAYLOR

FORENAME(S): COREY

Examination Answer Books

12 Pages

Instructions to Candidates

1. Please give the information requested in the boxes below.
2. Enter your name in the space provided to the right, fold over and seal down (using the label provided) to hide your name.
3. The answers must be legibly written. The Examiners can either ignore illegible script or have it typed and the expense of typing charged to the candidate.
4. Write your answers in ink.
5. Begin each answer on a fresh page. Do all rough work in the answer book and cross it through.
6. No part of this book is to be torn out; it must be handed in complete.
7. If more books than one are used, fasten them together using the tags available from the invigilator(s).
8. **Every book must be handed in. In no circumstances may answer books, used or unused, be taken out of the examination venue.**

1	Student Number	1708443/1
2	Title of Examination Paper	Micros, peripheral and interfaces.
3	Module Code	T.B.A.
4	Date and Time of Examination	P.M, 8/5/19
5	Examination Venue	MAIN HALL
6	Seat Number	
7	Number of Answer Books handed in	1

List the numbers of the questions you have attempted	Office Use
2	5 ✓
3	3 ✓
4	6 ✓
5	7 ✓
1	5 ✓
Office Use	26

(Gweler yr ochr arall am y fersiwn Cymraeg)

2) a) A PIC processor with 8 locations in a stack would affect the structure of the program written for it due to many possible memory locations & possibilities. This can affect speed of the program & slow down outcomes tasks.

b) - Subroutines are a specific part of code in the program used to call ^{certain} commands ^{or tasks} throughout the program. It is possible to know when these will occur as you would implement these to be called when needed. Once the subroutine has been completed, the rest of the program will continue to run.

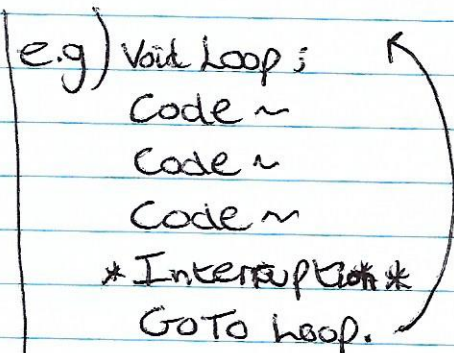
- Interrupt routines are different, in that they cannot be called and ~~can~~ can sometimes be unpredictable. An interrupt routine is triggered by ~~something~~ an occurrence in either hardware or software, such as something timing out, a limit switch or interlock, for example.

An interrupt routine will stop the continuation of the program until either the ~~aff~~ affecting component is reset, or until it is otherwise safe for it to continue.

2) (Cont.d)

c) - To resume a program operation in its correct state following an ~~it~~ interrupt, you would include a 'GoTo' command directly ~~to~~ after the interruption. This would enable the program to resume from the next step/command in the process.

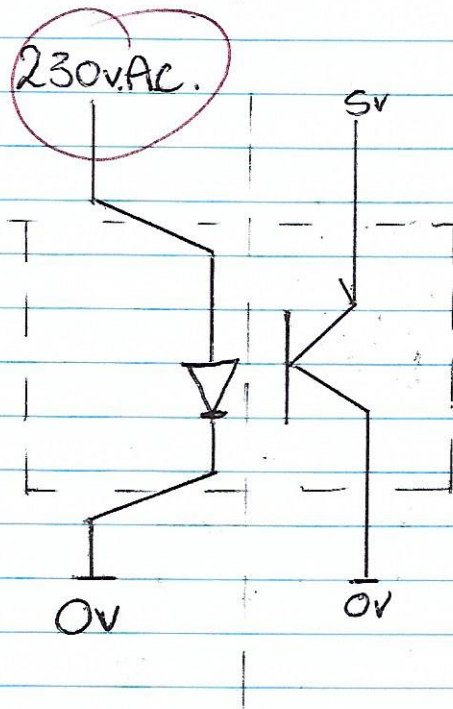
e.g) Void loop:
Code ~
Code ~
Code ~
Interrupt
GoTo loop.



3) a) - It is not possible to connect an a.c. load directly to the output of a 16F84A PIC, due to the inability of the PIC to handle the peak-to-peak charge of an a.c. voltage.

b) - A suitable solution for the above would be to use either a form of relay or an optocoupler in between the A.C. load and the 16F84A PIC. ✓

Optocoupler example:-

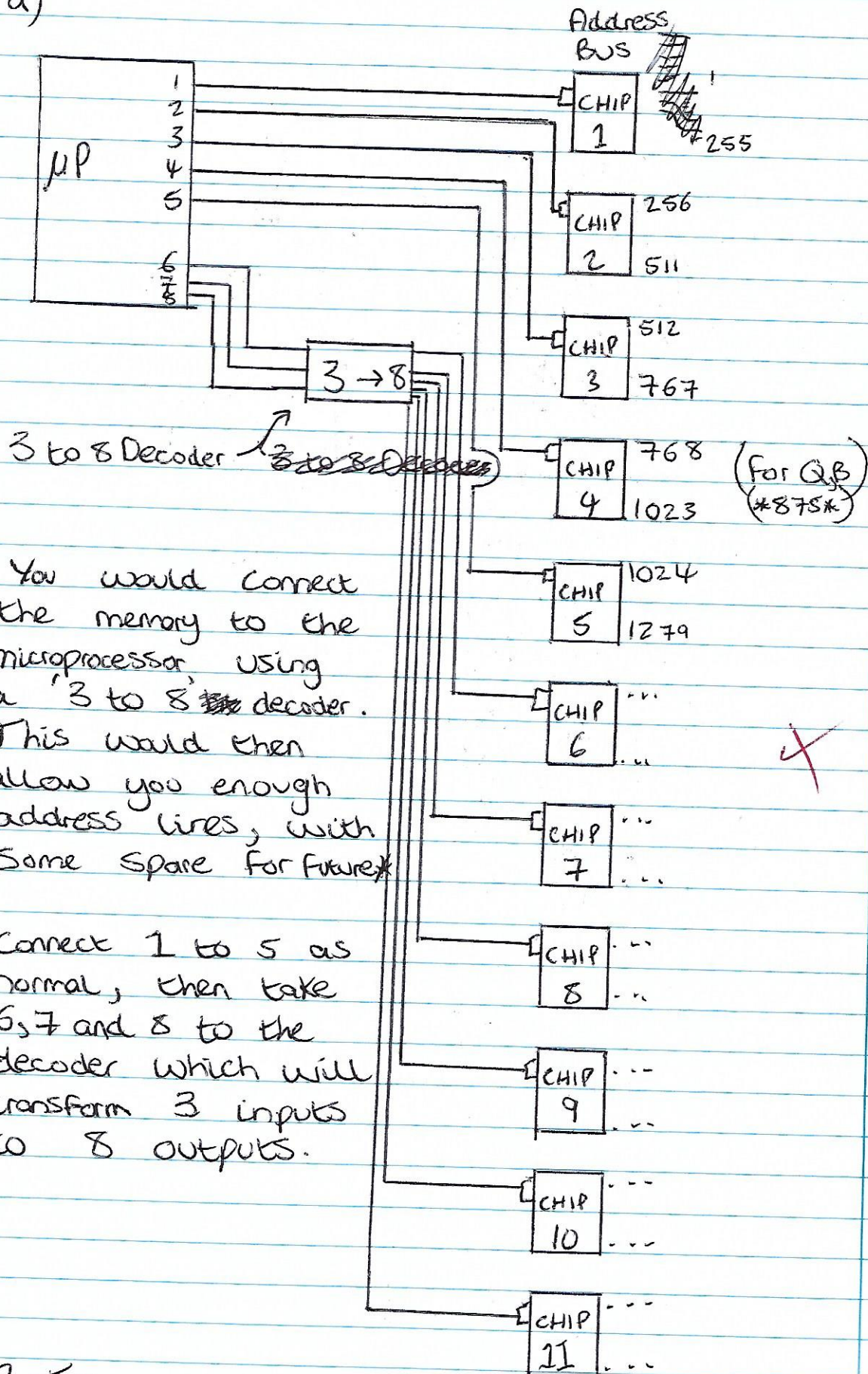


* This solution would provide a non-contact switch ~~light~~ (via light in this example), which would protect the 16F84A PIC's integrity whilst allowing it to control the 230V AC mains operated lamp. *

c)

d) You would do this by use of a relay or optocoupler. ✓

4) a)



* You would connect the memory to the microprocessor using a '3 to 8' decoder. This would then allow you enough address lines, with some spare for future.

* Connect 1 to 5 as normal, then take 6, 7 and 8 to the decoder which will transform 3 inputs to 8 outputs.

P.T.O.

4) (Cont.d)

a)

Table for

3 to 8

Decoder:-

	1024	512	256	128	64	32	16	8	4	2	1
10	9	8	7	6	5	4	3	2	1	0	
0	0	0	1	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	0	1	1
0	1	0	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1
1	0	1	1	1	0	1	1	1	1	1	1
1	1	0	1	0	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1

b) To access memory location '875' :-

Convert to binary:-

1024	512	256	128	64	32	16	8	4	2	1
0	1	1	0	1	1	0	1	0	1	1
Memory location/ Chip number					Location in chip.					

 $\therefore$ 875 is in chip no. 4.

$$\begin{array}{r}
 \text{To check} \quad \begin{array}{r} 128 \quad 64 \quad 32 \quad 16 \quad 8 \quad 4 \quad 2 \quad 1 \\ 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \end{array} = \begin{array}{r} 111 \\ 768 + 107 \\ = 875 \end{array}
 \end{array}$$

 $\therefore$ Chip No. 4, memory location 107.

5) a) 4 Key Differences between RS232 & SPI :-

SPI (Serial Peripheral Interface)	RS232 (Asynchronous)
High Speed, Low Distance	Lower Speed, High Distance.
Synchronous.	Asynchronous.
2 wires	3-4 wires.

* SPI can travel at a high speed but is generally used for shorter distances.
 RS232 is slower than SPI but used for long distances.

* SPI can be used with only 2-wires, whereas RS232 tends to use 3 or 4 wires.

P.T.O.

5) b) Inter-Integrated Controller (I^2C):-

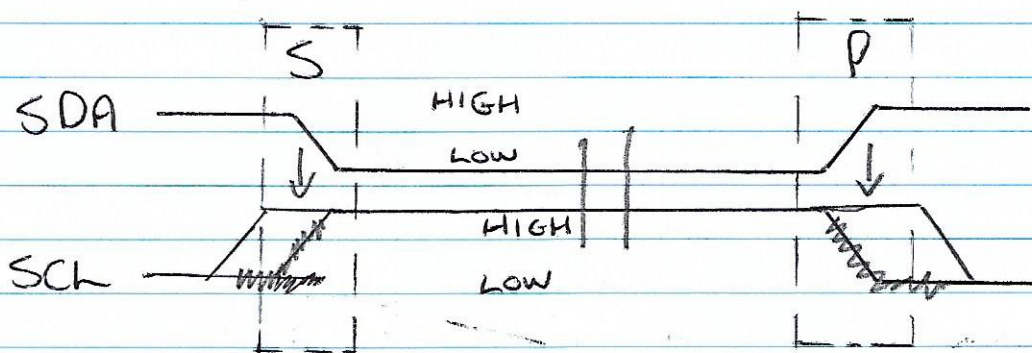
SDA - Serial Data wire/line.

SCL - Serial Clock line.

i) The master indicates a message is due to begin by using an 'S' or 'Start' command.

ii) - The master indicates a message has ended by using a 'P' or 'end' command.

*i/ii) - These are both seen by the behaviour on the two 'lines', being either high or 'low'.
e.g.) :-



iii) - A slave indicates to a master that it is not ready to receive data using an 'ACK' signal. (NOT ACKNOWLEDGE).
 (* IF the slave was ready, then it would use the 'ACK' signal. *)
 (ACKNOWLEDGE)

P.T.O.

5) b) iv) :-

Successful transfer :-

| S | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |

~~Slave 2~~

Slave 1 - Address - 0010101

Data - 00101011 +
001011

~~Address - 0010101
Data - 00101011~~

~~Address - 0010101
Data - 00101011~~ +

Slave 2 :-

Address - 00101111 +
Data - 00111111 +
1'0'1'0'1'1'1'1'0

Slave 3 :-

Address - 0011001 +
Data - 11101011 +
0'0'0'0'0'1'0'0

Full Sequence :-

| S | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

?

1)

a)

Post1 equ OE

MOVLW 0x58

MOVWF Post1

MOVLW 0x3A

ADDWF Post1,1

W

58

Post1

58

58

58

3A

3A + 58

Z = Zero

C = Carry

DC = 'little' carry

8 4 2 1

1 0 1 0

∴ 3A + 58

128 64 32 16 8 4 2 1

∴ 3A: 0 0 1 1 1 0 1 0

58: 0 1 0 1 1 0 0 0

1 0 0 1 0 0 1 0 ✓

Z = 0

C = 0

DC = 1 ✓

A = 10

B = 11

C = 12

D = 13

b)

Data1 equ 0x14

MOVLW 0x5A

MOVWF Data1

① RLF Data1

② INCF Data1

③ RLF Data1

④ INCF Data1

W

5A

Data1

5A ✓

5A ✓

5A ✓

5A ✓

5A ✓

5A ✓

5A ✓

5A ✓

① RLF 5A:-

$$= \boxed{0} 0 1 0 1 1 0 1 0 \text{ (SA)}$$
② INCF = Data1 + 1 ∴ $\boxed{0} 0 1 0 1 1 0 1 0$ = $\boxed{0} 0 1 0 1 1 0 1 1 = \text{(5B)}$

③

RLF 5B = $\boxed{0} 0 1 0 1 1 0 1 1$ $\boxed{1} 0 0 1 0 1 1 0 1$

8 4 2 1 8 4 2 1

(8D) 2 2

④ INCF(8D) = 00101110

1) b) (Cont. d)

∴ Contents of mem. location 14 = 2E

c) Count1 eqv 0x12

```
MOV LW 0x06  
MOVWF Count1  
CALL Delay1  
DECF Count1  
GOTO LOOP  
NOP  
NOP
```

[illegible]

Count1 equ 0x12

```
MOV LW 0x06  
MOVWF Count  
CALL Delay1  
DecF Count  
GOTO LOOP  
NOP  
NOP
```

OR

Count1 eqv 0x12

```
MOV LW 0x06  
MOV WF Count1  
CALL Delay1  
CALL Delay1  
CALL Delay1  
CALL Delay1  
CALL Delay1  
DecF Count1  
GOTO LOOP
```

* This is by adding codes, not altering existing lines. *