

12 Pages

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	160 33 21 / 1
2	Title of Examination Paper	COMPUTER ENG. + MICROPROCESSORS
3	Module Code	M4X 00370
4	Date and Time of Examination	9 th May 2018.
5	Examination Venue	MAIN HALL TYCOON.
6	Seat Number	4.
7	Number of Answer Books handed in	

[illegible]

1. A) equ OE
;

MOVLW 0x78 0111 1011

MOVWF Post 1

MOVLW 0x96 1001 0110

ADDWF Post 1,1

MOVLW 0x78

ADD 0x96

~~MOVLW 0x78~~

$$\begin{array}{r}
 \begin{array}{r}
 0111\ 1011 \\
 1'0'0'1\ 10'110 \\
 \hline
 110010001
 \end{array} \\
 \begin{array}{r}
 0111\ 1011 \\
 1'0'0'1\ 10'110 \\
 \hline
 100010001
 \end{array}
 \end{array}$$

Z = 0 as not all zeros

DC = 1 as carry over from 4th order

C = 1 as carry over from M.S.B (most significant bit)

OE = 100010001

③ ✓

B) equ 0x14
;

MOVLW 0x4C

0100 1100 into W reg.

MOVWF Data1

mov contents of W into Data1

RLF Data1

Rotate Left

INCF Data1

increment Data1 by +1

RLF Data1

Rotate Left

INCF Data1

increment Data1 by +1

carry → 0 0100 1100

0100 11000

increment by +1

carry → 0 0100 1101

0100 11010

increment by +1

0100 1110

equ 0x14 = 0100 1110 carry bit will be 0

① ✓

1. c) Count 1 equ 0x12 — assembly directive
 ;
 Loop MOVW 0x06 Put 6 into Wreg
 MOVWF COUNT 1 Put content of W into Count
 CALL DELAY 1 call delay
 DECF COUNT 1 decrement count 1 by 1
 GOTO LOOP goto start
 NOP
 NOP

1st have to change MOVW 0x06 to MOVLW 0x05 ✓
 Then DECF needs to be changed to
 DECFSZ to skip bsr goto when count = 0 ✓
 to go to NOP. so

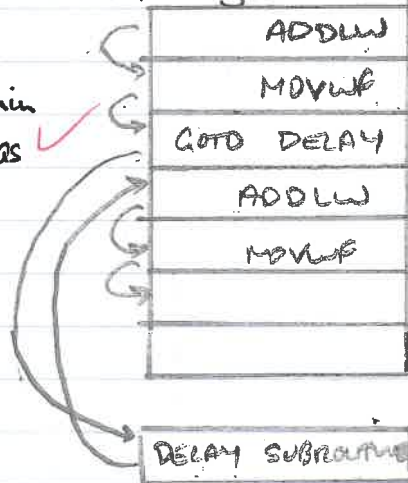
COUNT 1 equ 0x12
 Loop MOVW 0x05
 MOVWF COUNT 1
 CALL DELAY 1
 DECFSZ COUNT 1, 1
 GOTO LOOP
 NOP
 NOP

Also need a subroutine for Delay 1 and a return after
 it for it to return to the program.

- 2a) a subroutine - an example would be a delay whereby the subroutine decrements a pre defined value by 1 each time until zero is reached - it then skips the next command and RETURNS TO the following line of code from where it was called.

They can be called at anytime and require a RETURN at the end of it.

It is called for within the main program as a 'GOTO' or 'CALL'.



You can use RETFIE but and it will compile but result will be unpredictable so have to use RETURN.

The program counter is used and stores where it was in the assembly code.

Therefore the example would be a delay/countdown timer between traffic light sequences.

③ ✓

- b) Interrupts. An example would be a ~~zoo~~ pedestrian crossing at a set of lights. A global interrupt occurs when RBO (Port B, Pin 0) becomes high from an input. When the input has cleared the interrupt has finished the flag is cleared and RETFIE (return from interrupt) is called. The following instructions for the program counter to return to where it is called. The program counter then returns it to the origin of where it came.
- eg. push button is pressed at a crossing and the lights turn red. Once the ~~sensor~~ timer detects clears, the flag interrupt flag is cleared and RETFIE returns program to normal sequence.

All normal sequence is halted when an interrupt occurs.

③ ✓

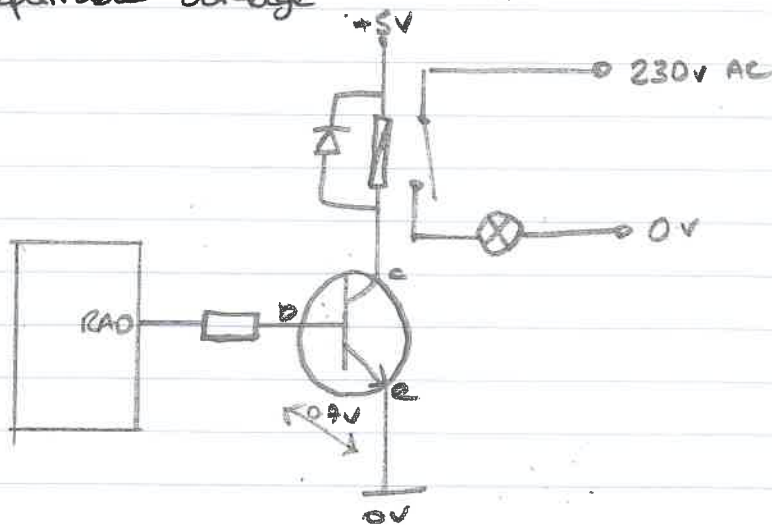
c) The stack. Stores the assembly

Saves the assembly code onto the top with a PUSH'd and when the interrupt finishes the ~~last~~ assembly code is retrieved using a POP'd. It is saved so that when the normal routine resumes all variables will be the same otherwise the program will be unpredictable as the W register will not have been saved. The program code always reads one line ahead of the assembly code.

d) When an interrupt occurs you should save the last line into a defined ^{part} ~~part~~ of the W register and when the interrupt finishes it should be retrieved. At the end of the interrupt RETFIE is placed to Return from interrupt.

- 3 a) The PIC16F84A is not able to control 240V ac. nominal voltage is approx 5V DC - this would result in irreparable damage.

b)

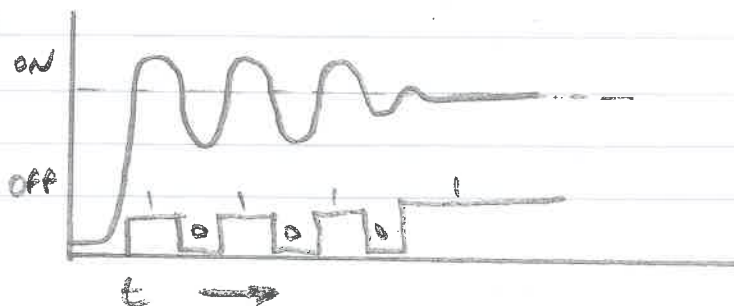


As a pic can only supply 25mA as an output a FET (field effect transistor) will switch operate a coil on a relay. We need a diode ~~over~~ across the coil otherwise the transistor will be permanently damaged. Due to Lenz's law: a change in current from a magnetic field will oppose the direction of the flux and the mechanical force ~~is~~ applied to

Once voltage is applied to the base of the transistor the collector coil which is attached to the collector leg energises and the ~~relay close~~ contact close on the electrically isolated 230V side, completing the circuit allowing the lamp to illuminate.

You could also set up Port B as an input to control the output of Port A to the transistor.

- c) Precautions to take. One precaution would ~~be~~ be to use a debounce. When a switch closes this happens



Instruct the BTFSS or input to read the input.

→ store this

→ call a delay (possibly 20 ms)

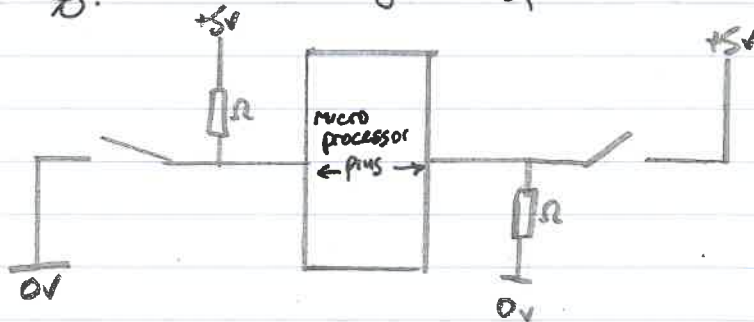
→ reread this input state

→ compare to existing

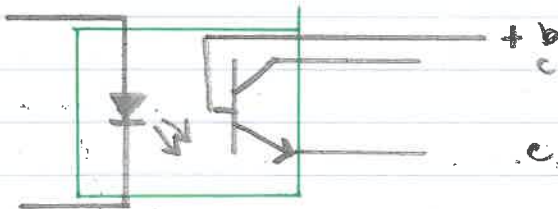
→ resume program

otherwise the bouncing will give inaccurate readings as it may see it as off

You could also use pull downs to avoid any floating voltages which would give spurious readings eg.



d) by using an optocoupler which would galvanically isolate it.

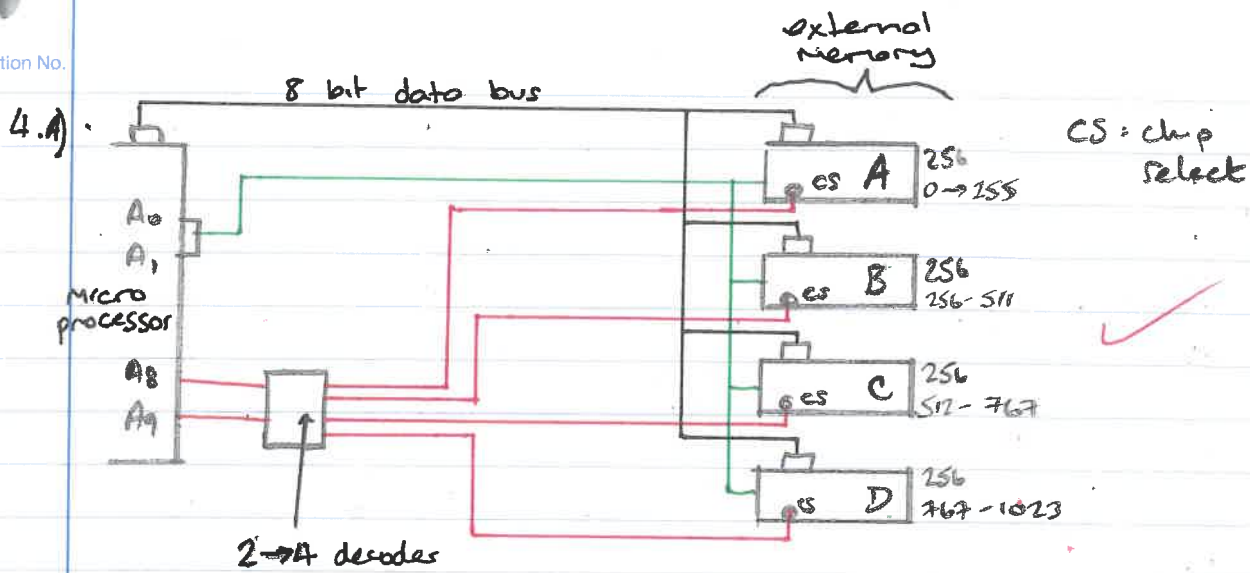


the infra red light emitting diode is not connected to the transistor.

It is separated by resin, a vacuum or another inert material, to make it intrinsically safe.

The semi conductive photo sensitive transistor is responsive to the I.R light emitted by the IR LED and allows voltage through when excited in a switched on state.

The collector of the leg would be attached to the input pin of the PIC microprocessor.



The 2 → 4 decoder turns off the output to the external memory bank that is required and turns on to those that are not. to the chip selects.

b) to address memory location 800
in binary 800 = 1 1 0 0 1 0 1 0 1 0

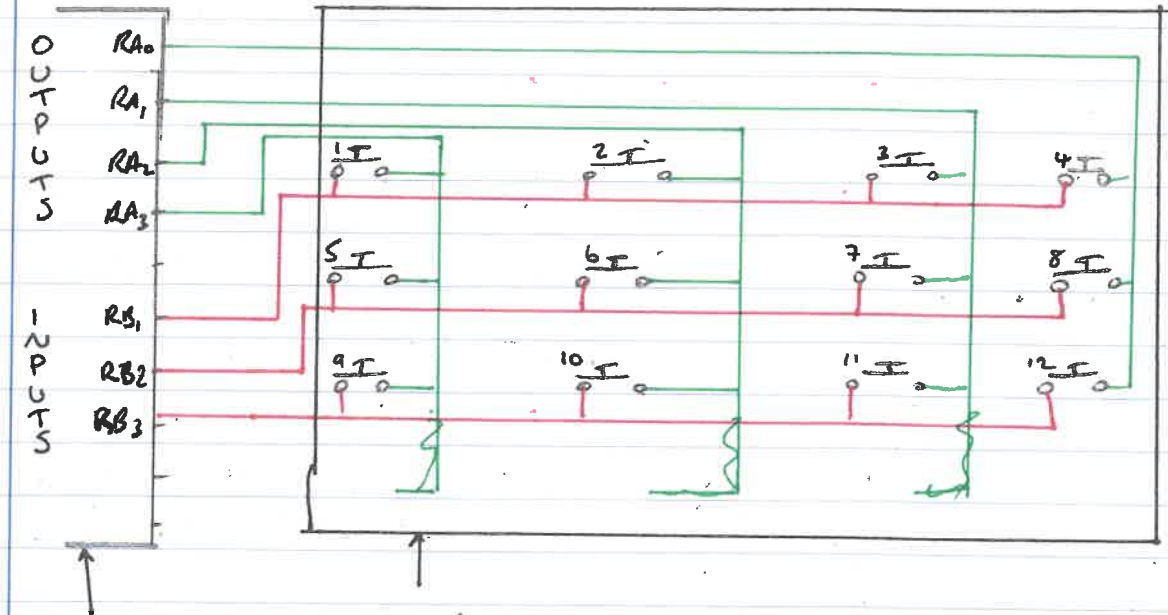
A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	1	0	0	1	0	1	0	1	0

chip decoder would be

1	A_8	A	1
		B	1
1	A_9	C	1
		D	0

The output to chip select on D is 100 so this is selected. as memory location 800 is in Bank D.

S.A)



microprocessor

keypad.

All switches are normally open.



5

- B) ~~The outputs are read one at a time and each~~
 The outputs (columns) are read one at a time on the inputs RB_1 through to RB_3 .
 When this completes the following column is set to high to be read RA_1 , and then $RB_1 \rightarrow RB_3$ are read in turn.

eg.

OUTPUTS

INPUTS.

	RA_0	RA_1	RA_2	RA_3	RB_1	RB_2	RB_3
line 1	1	0	0	0	0	0	0
line 2	0	1	0	0	0	0	0
line 3	0	0	1	0	0	0	0
line 4	0	0	0	1	0	0	0

Returns to line 1.

Should a button be pressed eg No 6 then this is how it would read it

OUTPUTS				INPUTS		
RA ₀	RA ₁	RA ₂	RA ₃	RB ₁	RB ₂	RB ₃
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	1	0	0	1	0
0	0	0	1	0	0	0

In this instance the output of RA₂ would be seen on the input of Port B pin: RB₂

Due to the high speed of the PIC 16F84A at 4MHz the time taken to press a button is much slower than the time it takes to cycle around so it will not be missed.

Port B can be set up as a BTFSS or BTFSC depending how you want it to work so that the value can be added to the W register or any other location to perform its calculation.