EICoM
ELECTRICAL COMPUTER MECHATRONICS

مختبر الانظمة المضمنة (خارجي)

للطالبة المبدعة

نتالي الكايد

إرادة - ثقة - تغيير

LAB (1) :- introduction to PIC microcontroller.

↘ microcontroller :- is a compact microcomputer or computer on chip

↘ It's a single IC that contains :-
1. Processor core
2. RAM
3. ROM
4. I/O control unit.

↘ it is designed for embedded applications.

↘ the main difference between a microcontroller and microprocessor is that a microcontroller has and internal bus with built in memory and peripherals.

↘ difference between microcontroller and computer is [specification not the modules].

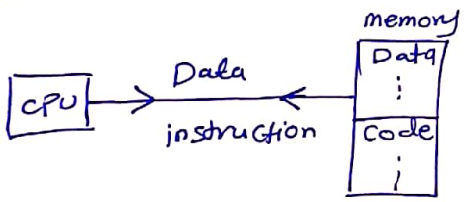
	Computer	MCU
RAM	8 GB	128 KB - 512 KB
Clock rate	1-4 GHZ	32KHZ - 20MHZ
CPU cores	up to 16	1
serial ports	USB, RS422...	USB, SPI,..
current drain	1.5 AmPs	50 mA

CamScanner

microcontroller architecture

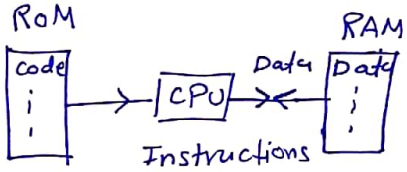
1 Von Neumann archi.—

- the system has single memory, so it has a one bus for data and instruction.
- 2 clock cycles needed to complete an istruction.
- Pipelining is not possible in this archi.—.



2 harvard archi.—

- 2 separate memories (storing program)
- instruction needs 1 cycle if pipelining strategies are implemented.



* What is the (PIC) microcontroller ?

PIC :- Programmable interface controller , and it's developed by micro chip technology ← الذكي الذي يتبعها Harvard archi.—.

1. Serial communication modules: مثل [on-board Peripheral] * فيها الصيد من الـ
2. UARTS.
3. motor contrd Kernels.

* بنقدر نصيد برمجتها لأنها بتستخدم الفلاش ميموري.
* بنقدر نبرمجها باستخدام لغة الأسيمبلي أو لغة C مثلاً.
* بجهاز اللاب رح نستخدم نوع PIC 16F84A.

PIC 16F84A features :-

① (35 instruction) عدد الانستركشن اللي فيها

② بتشتغل على تردد 20 MHz

③ 1024 word يساوي Program memory طول الـ

④ 14 bit طول الانستركشن يساوي

⑤ 8 bit يساوي طول الـ data bytes

⑥ 15 special function hardware registers فيها

⑦ 8 Level deep hardware stack. فيها

⑧ 13 I/O Pins (باتجاهين) فيها (بمعنى ممكن تشغل منها ~~حاجات~~)

⑨ بتقدر تبرمجها أكتر من عشر آلاف مرة

⑩ تستخدم (power) قليلة جداً، وتعتبر high speed technology

⑪ تقسيم الـ Ports فيها كالتالي :-

$(R_{A0} - R_{A3}) = A$

$(R_{B1} - R_{B7}) = B$

* Memory Map :- it shows all the available regs in data memory

* the regs are organized as a table format and has 2 part :-

1. Upper Part :

(special function Reg) بنسميها

تستخدم للتحكم بالـ PIC operation.

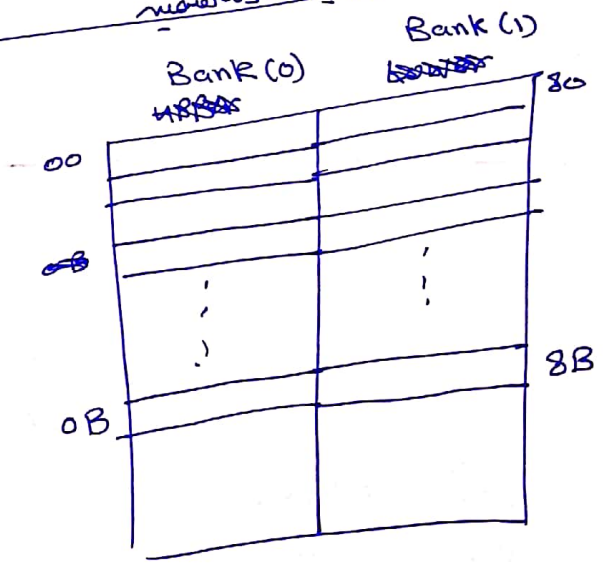
2. Lower Part :-

~~تخزين~~ (general function Reg) بنسميها

data memory locations عبارة عن

that the user is free to use as he wishes.

الميموري مقسومة لقسمين



لما نركب الكود بنكون في bank (0).

* الـ CPU الخاصة بـ PIC 16F84A بتتكون من ALU / Decoder / Control unit.

* الـ CPU هاي بتدعم الـ (RISC archi...) يعني بتدعم عدد أقل من الانستركشن، بتعتمد على الانستركشز الأساسية

* الـ instruction time مساوي تقريباً لكل الانستركشن اللي فيها.

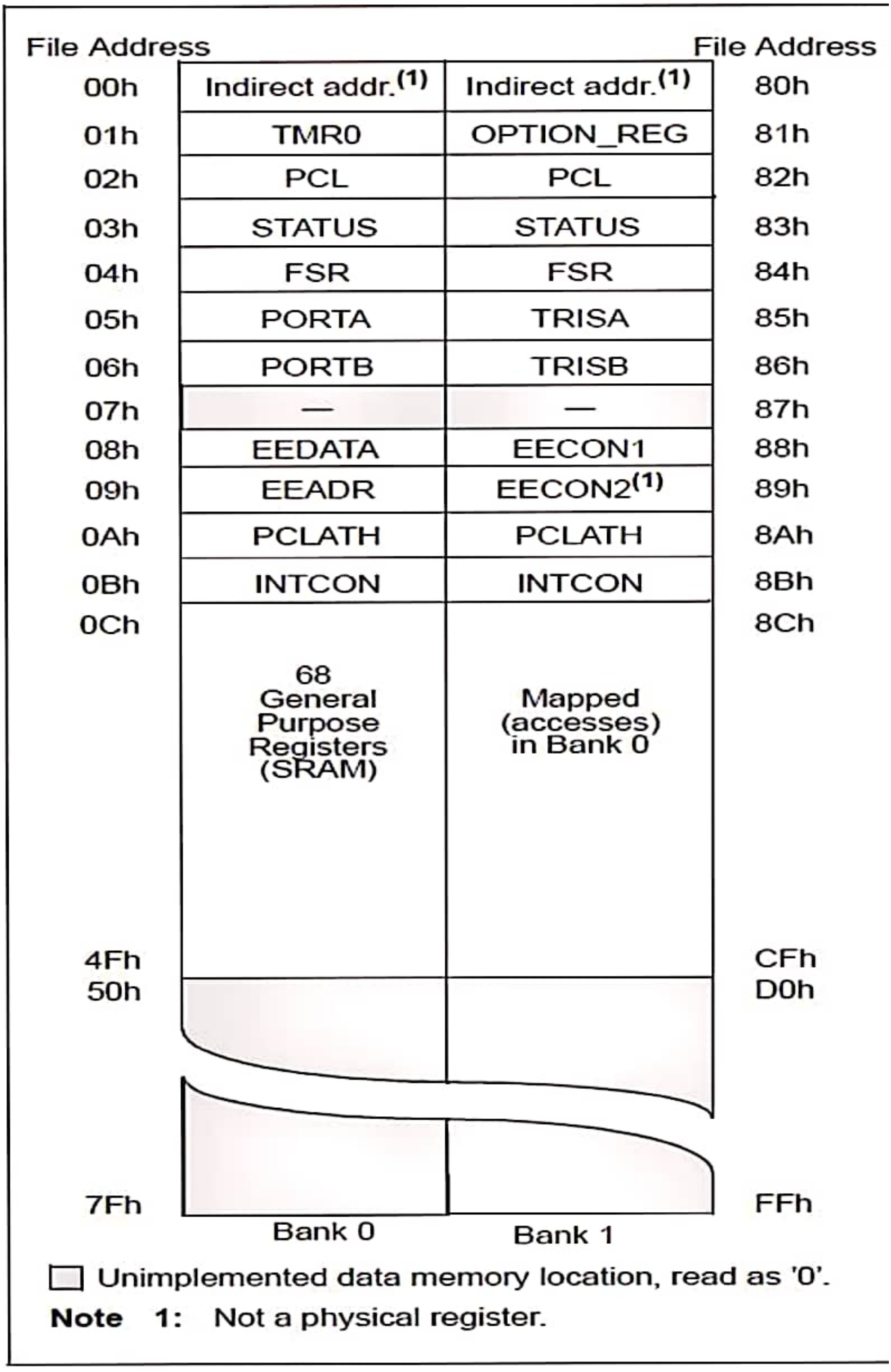


Figure 1: Data memory organization

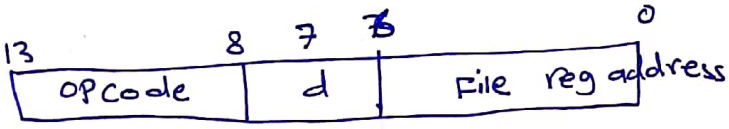
CamScanner

4 What is the Bank for ?

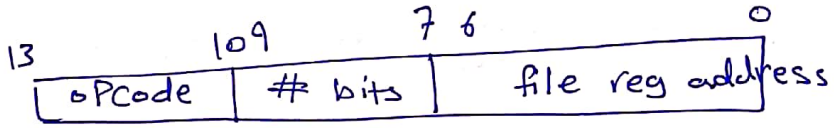
* 14 bit instruction → use 7 bits to address location.
* memory space is organized in 128 B bank.
* Bank (1) is used to contrd the acutal operation of the PIC.
* Bank (0) is used to manipulate the data.

* Instruction Set :- فيها ٣ أنواع

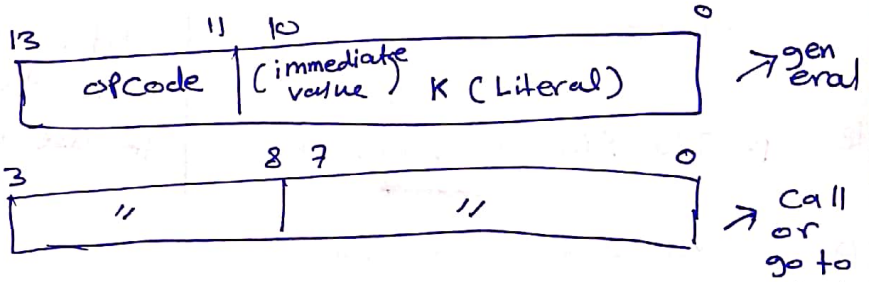
→ Byte Oriented



→ Bit oriented



→ Literal and control



---

* ٢ / ندرس هدا أول ٣ انتركشز :-

Movement instructions

moving constants the working Register (Loading)

↓

اسمه

MOVLW ~~ الرقم

moving values from the data memory to the working Reg (Loading)

↓

اسمه

Movf ~~ الادريس

moving values from the working Reg to the data memory (storing)

↓

اسمه

Movwf ~~ الادريس

CamScanner

:-

MovLW 05 :- moves the constant 5 to working Reg.

MovLW 10 :- moves " " 16 to " "

هاذ default decimal

* MovLW d'10' :- " " decimal value 10 to " "

* MovLW .10 :- " " " " " " " "

* MovLW b'1001110' :- " " Binary " " " "

* MovLW O'76' :- " " octal " " "

* MovLW A'g' :- " " ASCII " " "

* MovLW 0XAB :- " " constant AB_h " "

لازم نحطها بيش بحرف

hexa decimal

* MovLW H'7F' :- " " " $7F_h$ " "

* MovLW CD :- غلط لأنه الرقم بيش بحرف فلازم أحط 0X قبله ، وطبعاً الديفولت هو بنظام الهيكسا.

أمثلة :-

* MovWF 01 :- copies the value found in W to TMR0

* MovWF 05 :- " " " to PORTA.

* MovWF 0C :- " " " to GPR (location 0C)

* MovWF 32 :- " " " to GPR (location 32)

في الأمثلة السابقة بيبين انه إلقدرين او هي TMR0

* MovWF 52 :- غلط لأنه خارج رينج الداتا ميموري GPR (0C - 4F) ، الرينج فقط (8C - CF)

أمثلة :-

* MovF 05,0 :- copies the content of PORTA to the working reg.
* MovF 2D,0 :- " " --- GPR 2D the - - -
* MovF 05,1 :- copies the content of PORTA to itself

هاد فائدة انه بنقل للعالق بحيث او بحيث activate statuse

* MovF 2D,1 :- copies the constent of 2D to itself.

---

* MPlab * الشغل على البرنامج

1. Creat folder on any drive.
2. Creat AsM file in the folder.
3. open MPLAB Program.
4. PIC16F ← PIC اسم بنبدء
5. Project Name, browse, add file, Next, finish.
6.

مثال :-

```
MovLw 5
MovWF 1
MovLw 2
MovWF 0B
MovF 01,0
NOP
END
```

لعني working Reg

كيف نشوف run

6. debugger → select Tool → MPLAB SIM.
7. عشان نشوف النتائج → view → watch
   → " → File Register

TABLE 7-2: PIC16CXXX INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb	LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS							
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff		C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff		Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff		Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx		Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff		Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff		Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1 (2)	00 1011 dfff ffff			1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff		Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1 (2)	00 1111 dfff ffff			1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff		Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff		Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff			
NOP	-	No Operation	1	00 0000 0xx0 0000			
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff		C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff		C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff		C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff			1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff		Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS							
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff			1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff			1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff			3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff			3
LITERAL AND CONTROL OPERATIONS							
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk		C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk		Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk			
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100		$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk			
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk		Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk			
RETFIE	-	Return from interrupt	2	00 0000 0000 1001			
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk			
RETURN	-	Return from Subroutine	2	00 0000 0000 1000			
SLEEP	-	Go into standby mode	1	00 0000 0110 0011		$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk		C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk		Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

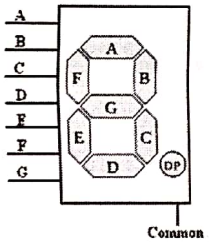
2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 Module.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

CamScanner

Command	Binary D7	D6	D5	D4	D3	D2	D1	D0	Hex
Clear Display	0	0	0	0	0	0	0	1	01
Display & Cursor Home	0	0	0	0	0	0	1	x	02 or 03
Character Entry Mode	0	0	0	0	0	1	1/D	S	04 to 07
Display On/Off & Cursor	0	0	0	0	1	D	U	B	08 to 0F
Display/Cursor Shift	0	0	0	1	D/C	R/L	x	x	10 to 1F
Function Set	0	0	1	8/4	2/1	10/7	x	x	20 to 3F
Set CGRAM Address	0	1	A	A	A	A	A	A	40 to 7F
Set Display Address	1	A	A	A	A	A	A	A	80 to FF

1/D: 1=Increment*, 0=Decrement
S: 1=Display shift on, 0=Off*
D: 1=Display on, 0=Off*
U: 1=Cursor underline on, 0=Off*
B: 1=Cursor blink on, 0=Off*
D/C: 1=Display shift, 0=Cursor move
R/L: 1=Right shift, 0=Left shift
8/4: 1=8-bit interface*, 0=4-bit interface
2/1: 1=2 line mode, 0=1 line mode*
10/7: 1=5x10 dot format, 0=5x7 dot format*
x = Don't care
* = Initialization settings



STATUS REGISTER

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC(1)	C(1)
bit 7							bit 0

INTCON REGISTER (ADDRESS 0Bh, 8Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

OPTION REGISTER (ADDRESS 81h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

CamScanner

Subject Directives. Date No.

Directives :- it isn't an instruction, it is assembly ~~language~~ commands that appear in the source code but aren't usually translated directly into opcode.

وظيفتها انها تتحكم بال (assembler) بتظهر بال source code وبتختلف باختلاف ال assembler المستخدم machine code اكتر ما بتتحول لـ

ال assembler المستخدم بالباب هو [MPASM].

* directives (Like the PIC instructions) aren't case sensetive.

[1] END directive :- لازم تكتبه في آخر سطر من الكود لأنه بدل على نهاية البرنامج

[2] Equate directive :- بنستخدمه عشان نعرّف أو اسم لـ constant أو اسم لـ memory address bit لريجستر.

مثال :- x EQU 5 → X=5

Tmp EQU 20

Timer0 EQU 01 → آدرس التايمر

intcon EQU 0B

workrg EQU 0

END

movLw 5 → انقل الرقم 5 إلى working Reg.

movwF Timer0 → انقل من الـ wR إلى الملف اللي اسمه Timer0 وممكن أكتب 01 لأنه 01 يعبر عن آدرس التايمر

movLw 2 → خزن رقم 2 في wR

movwF intcon → خزن من wR إلى intcon

movF Timer0, workrg → نقل القيمة من Timer0 إلى wR

NOP

END

NOTEBOOK

CamScanner

Subject Date No.

	address	value	hex	Dec	binary
النتائج :- X =					
WR =	ريجستر محجوز	0X05	0X05	5	0000 0101
Timer0 =	01	0X05	0X05	5	0000 0101
intcon =	00	0X02	0X02	2	0000 0010
Tmp =	10	0X00	0X00	0	0000 0000

* لو نقلنا نفس الكود بكن عدلنا على أول سطر وهو حمار ←

movLw X

بما انه قيمتها 5 فما ← 2 تتغير اشي على النتائج .

بما انه كاتبين L فهو عامل ← Constant على أنها X

[redacted]

هاي الـ data memory بتتكون من جزأين ← upper Part, Lower Part

Special function Regs فيه الـ upper P. الـ

	Bank(0)	Bank(1)	
00h	TMR0	OPTION-REG	80h
	PCL	PCL	
	STATUS	STATUS	
	FSR	FSR	
	PORTA	TRISA	
06	PORTB	TRISB	86
	—	—	
08	EEDATA	EECON1	88
	EEADR	EECON2	
	PCLATH	PCLATH	
0C	INTCON	INTCON	8C

بدك ما اتعامل مع اللوكيشن اه يلي فيه الريجستر TMR0 ، فالأسهل عليي اني اتعامل مع اسم الريجستر بدل عنوانه .

CamScanner

Subject Date No.

```
include "PIC 6F84A.INC"
MovLW 8
MovWF TMR0
NOP
END
```

النتائج:-

TMR0 = 8

WR = 8

مثال:-

```
include "PIC 6F84A.INC"
MovLW 8
MovWF TMR0
MovWF TRIS A
```

TMR0 = 8

WR = 8

TRIS A = IF

ليش؟؟

TRIS A موجودة في bank(1) واحنا صون بنشتغل على bank(0) ، فعشان أغير البانك لي

عم بشتغل عليه ، لازم أعمل switch ، بعمل switch بطريقتين:

① directive (bank select) :-

يعني بالكود كالتالي

```
include "PIC 6F84A.INC"
MovLW 8
BANKSEL TRIS A
MovWF TRISA
```

TRIS A = 8

② by using bcf , bsf instructions:-

BCF STATUS , PR0 ← PR0 in STATUS is 0 → switch to bank 0

BSF STATUS , PR0 ← PR0 in STATUS is 1 → switch to bank 1

CamScanner

Subject Date No.

3 Cblock :- is used to define variables in the data memory.

Cblock هو directive بتجربه عشان أعرف مجموعة متغيرات بدل ما استخدم EQU ، وبعرّف المتغيرات بـ [sequential location] ، كالتالي :-

```
cblock 0X35
varx
vary
varz
endc
```

مثال :-

```
INCLUDE "P16F84A.INC"
CBlock 0X20
VAR X
VAR Y
ENDC
MovLw 8
MovWF VARx      ← ح يخزن قيمة w إلى VARX
CLRF VARY       ← ح يعطي قيمة صفر للمتغير VARY
```

* FLAGS :- في عنا 3 flags بالـ PIC اللي بنستعملها دايمة

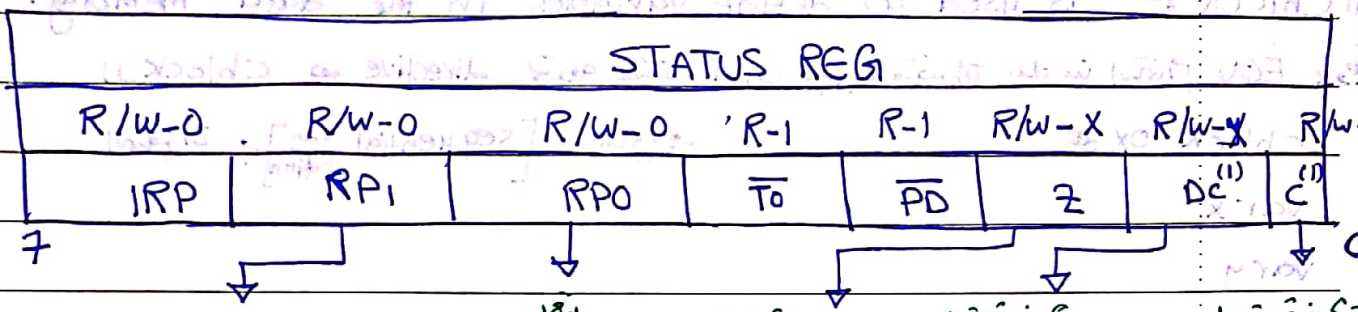
(Z) zero (DC) Digit carry carry (C)

بعض الانستركشن بتأثر على قيم الـ flags ، وبعضهم ما بأثر.

مثل MovF بتأثر على Z ، مثل movLw, MovWF

CamScanner

Subject Date No.



هاي بستخدماتي PIC16F877 لي فيها 4 bank عشان أنتقل بينهم بغير قيمة فلو بدي أرجع على bank 3 بخلي قيمته = 11

بتنقل bank switching فلو بدي أنتقل إلى bank1 لازم أخلي قيمة 1 واذا بدي أرجع على bank0 بخلي قيمة 0

بتكون قيمته 1 اذا الناتج يساوي صفر

بتكون قيمة 1 في حال وجود من carry first Nibble of the result

بتكون قيمته 1 في حال وجود من carry most sig.-- bit of the result.

* Single bit main pulation :- هاي اله instruction اللي بتعدل على (1 bit) داخل ريجستر وهي 2 ← bcf / bsf

(1) bcf f,b
bit clear file ← address of file ← number of bit.
b=0-7

مثال :- bcf PORTB , 0] clear bit 0 of port B.
يعني خلي قيمة صفر
PORTB في bit0 ال Reg

(2) bsf f,b
set bit b of Reg f , b=0-7

مثل :- bsf STATUS , C] set the carry bit
يعني خلي قيمته 1

NOTEBOOK

Subject Date No.

* Logical and arithmetic instructions :-

ADD , SUB , AND , XOR , IOR (the ordinary inclusive OR) .

(1) ADD :-

↘ ADDWF f,d ⇒ add W and F

↘ ADDLW k ⇒ add Literal and W

مثال :-

```
cblock 0X20
    TEMP1
    endc
    addLW 5        ⇒ W = 5 + W
    addwF TEMP1, F ⇒ TEMP1 = TEMP1 + W
```

بما انه صون مكتوب F معناته ناتج الجمع يرجع على TEMP ، لكن اذا كان مكتوب W أو 0 معناته النتيجة ترجع على W .

مثال :-

```
INCLUD" P16F84A.INC"
cBlock 0X20
VAR X
VAR y
TEMP1
ENDC
MovLW 8
addLW 5  → W=W+5
addwF TEMP1, 1 → TEMP= TEMP+W
MovWF VARx
CLRF VARy
```

WR = 13
TEMP = 3 قبل
TEMP = 16 بعد

CamScanner

Subject Date No.

(2) SUB :-

- SUBWF f,d → subtract w from F
- SUBLW K → subtract w from Literal

مثال :- subLW 5 → w = 5 - w

subwF TEMP1 , F → TEMP1 = TEMP1 - w

اذا كانت F أو 1 ← خزن النتيجة في TEMP

اذا كانت 0 أو w ← خزن النتيجة في WR

(3) Logical operation :-

مثال :- andLW B'00000111' → force upper 5 bits of w to Zero

andwF TEMP1, F → TEMP1 = TEMP AND W

andwF TEMP1, w → W = TEMP1 AND W

iorLw B'00000111' → force lower 3 bits of w to 1

iorwF TEMP1, F → TEMP1 = TEMP1 OR W

XorLw B'00000111' → Complement lower 3 bits of w

XorWF TEMP1, W → W = TEMP1 ⊕ W

CamScanner

Subject Date No.

مثال:- implement the following function = (X+Y) xor Z

```
include"P16F84A.inc"
Cblock  0x25
Varx
Vary
VarZ
Result
endc
    org   0x00
main
    movLw B'01010101'        اللي بعمل انه بخزن القيمة أولاً في الـ WR ثم بنقلها إلى المكان المطلوب
    moviwf Varx                                                    المكان
    movLw B'00110011'
    movWF Vary
    movLw B'00001111'
    movWF VarZ

    movf  Varx,w             هنا بجي أعمل العملية المنطقية
    iorwf Vary,w             لازم يكون قيمة موجودة في WR والثانية
    xorwF VarZ,w             في ملف ، عشان هيك نقلنا قيمة Varx إلى WR.
    movwF Result             وخزنا ناتج الـ or في w عشان نعمل العملية
    no P                     يلي بعده وهي xor .
    END
```

NOTEBOOK

CamScanner

Subject Date No.

Clear / Swap.

- ~~Clear~~
- CLRW → بتصفير قيمة WR=0 ، يؤثر على Z
- CLRF F → بتصفير قيمة f = 0 ، يؤثر على Zero flag وبتصفير قيمة واحد.
- SWAPF F,d → بعمل تبديل للقيم بين الـ lower nibble والـ upper nibble في الريجستر

Increment / decrement / Complement.

- INCF f, F(W) → زيد على F وخزن الناتج في F أو W
- DECF f, F(W) → // // // // نقص من F
- COMF f, F(W) → // // // Complement اعمل

Rotate.

- Rotate Left → RLF f, F(W)



- Rotate Right → RRF f, F(W)



CamScanner

Subject Date No.

مثال :- cBlock 0X35

X1
X2
X3
ENDC
clrw
clrf X1
incf X1 ,1 → لو كان مكتوب 0 أو 2 تتخزن النتائج في w
incf X2 ,0
comf X2 ,w
decf X1 , 0 → بتخزن النتائج في w
rrf X3 , 1 → بتخزن النتائج في X3
NOP
END

X1 = 1
X2 = 3
X3 = 9
w = 0

* Conditional Branch :- الـ PIC عاد فيه 4 انتركشنز كونديشنال

btfsc ، btfss ، decfsz ، incfsz

btfsc f,b
btfss f,b
decfsz f,d
incfsz f,d

btfsc: abit is clear : 0 (bit test file, skip if clear) — بتفحص bit معينة في الريجستر عن skip لو كانت زيرو بتعمل وبال next instruction الـ بتكمل الكود.

btfss: a bit is set one : 1 (bit test file, skip if set) — بتعمل عكس الأولى

decfsz: zero flag after dec the register — بتعمل decre.. وبعدين بتشوف الـ zeroflag وإذا قيمتها صفر بتعمل Skip

incfsz: // — نفس الشيء بس incre..

NOTSkip BIT OK

Subject Date No.

* بنقدر نعمل تشييك على قيمة ال flags اذا كانوا 0 أو 1 من خلال:

btfss/ btfsc STATUS ,C

btfss/ btfsc STATUS ,Z

Subject Date No.

* هلأ بدنا نطلع الـ machine code manually ...

مثال:- K ⟵ عبارة عن كونستنت طوله 8 bit

MOVLW 0X09 ⟹ opcode = 11 0000 0000 1001 = 3009 H
K 9

BTFSC VARX,4 ⟹ opcode = 01 1010 0010 0000 = 1A20 H
بدنا 3 bit بقولها إنك صار المثال موجود في آدرس معين b آدرس VARX

MOVF VARX,0 ⟹ opcode = 00 1000 0010 0000 = 0820 H
d آدرس VARX

* MOVLW K ⟹ 11 00xx KKKK KKKK
* BTFSC f,b ⟹ 01 10bb bfff ffff
* MOVF f,d ⟹ 00 1000 dfff ffff

هاي بتمثل Reg file address اللي range (0-7F)

دال (b) عبارة عن bit address بتراوح بين (0-7) يعني الرقم بتكون مثل ٣ مثلاً

d : destenation Sellect
غالباً بتكون قيمتها إما زيرو أو ١
↓
بتخزن الناتج في W

بتخزن الناتج في f

NOTEBOOK

CamScanner

Subject Date No.

* Go to / call

Go to: بنقلني إلى أي انستركشن.

call: بستخدمه لاستدعاء Subroutine. بحيث أقدر بتخزن قيمة PC في الستاك.

* Return / Return from interrupt:

بترجعني من الـ Subroutine وبتعمل pop للـ ~~next instruction~~ العنوان المخزن في الستاك.

- return → pop return address off of stack.
- ret LW 5 → pop return address (w=5).
- retfie → pop return address, re-enable interrupts.

special return: عبارة عن بستخدموا مع lock up table بحيث بتحفظ قيمة الـ w وبعديه بتعمل return.

بترجع من الـ interrupt service routin وبتعمل re-enable للـ interrupt

CamScanner

Subject Date No.

look up table :- وطريقة استدعاءها Subroutin هي نوع من أنواع الـ
بال call لكن إلها special return وهي retlw ، بترجع قيمة حسب الأبشن اللي بتوصلها.

+ a look up table is a block of data that is held in the program memory and which can be accessed by the program and used within it.

مثال :-
```
MovLw 3          → قبل البدء واستدعاء الفنكشن w=3
CALL MyLookUpTable   ع نفرض انه الفنكشن بيدأ تنفيذه
                     عند الأدريس 100.
:
100  MyLookUp TAble
     addwf Pcl,F     → ع يخزن قيمة الـ Pcl اللي إلها next instruction
                       ما تنفذ إلو (Pcl=Pcl+w)
101  nop
102  retLw 0x0A
103  retlw b'1001110'
104  retLw 23
105  retlw 18
106  retlw 54
```

Pcl = 101 + 3
= 104

هون بعمل return وبرجع للبرنامج الأصلي

	index	value
	1	0A
	2	9E
*	3	23
	4	18
	5	54

CamScanner

Subject Date No.

مثال :-

```
#include P16F84a.inc
CBlock 0X20
VAR x
Result
ENDC

movLW 3
Call mylookTable
movwF result
goto finish
mylookTable
addwf pcl, F
nop
retlw 0X0A
retLw 9E
retLw 23
retLw 18
retLw 54
```

لازم اتأكد منه

W = 35
result = 35

~~View~~ → Program memory : عن طريق البرنامج machin الـ code بنقدر نطلع

CamScanner

Subject Date No.

Q1:- write a code to find the number of ones in the variable x, and store the results in Res variable.

أمثلة اضافية
الحل حلّين وغير أكيد انه صحيح

```
CBlock 0x30
counter
x
result
F
endc
movLw x
movwF
L: decfsz F, 1
goto V1
goto finish
V1:
btfss X, F
goto L
V2:
MovLw counter
addlw 1
Movw counter
MOVlw counter Result
finish
Done
```

btfss f, F
go to L
go to V1

أو

```
movwF x
movLw 1
movwF F
movLw 0
movwF Counter
btFsc x, F
movLw 1
movwF counter
L: movLw 1
addwF F, F
movLw 0
btFsc x, F
movLw 1
addwF counter, 1
movLw 8
subwF F, w
btFsc status, Z
goto finish
movLw 1
goto L
```

NOTEBOOK

Subject Date No.

مثال :- write a code to express this function $f = (X \text{ or } y) \oplus (x \text{ and } z)$
واكتبه في

```
include "P16f877.inc"
cblock 0x30
X
n
y
z
r
endc
org 0x00
main
movf X , 0
iorwf y,0
movwf n
movf X , 0
andwf Z,1
movf Z ,0
xorwf n,0
movwf r
end
```

الكود صحيح وشغال ✓

CamScanner

Subject Date No.

مثال إضافي :- write a code that express the function R = X+Y+Z .

الكود صحيح وشغال ✓

```
cblock 0x20
X
y
Z
R
endc
MovF x,0
addwF y,0
addwF Z,0
MovwF R
END
```

مثال إضافي :- write a code to multiply any number by 7 (Perform the multiply as repeated addition)

```
cblock 0x20
X
count
endc
clrw
movLw 6
movwf count
movf X,w
loop
addwf X,f
decfsz count,f
goto Loop
goto finish
finish:
end
```

Subject Experiment(2) :- Date No.

I/O interfacing and the 7-Segment display.

* الـ PIC يلي هنتعامل معهم هما PIC 16F877 / PIC 16F84A :-

PIC 16F877 ← بتحتوي على عدد من الـ Ports

PIC 16F84A ← بتحتوي على 2Ports

- ✓ Port (A)
- ✓ Port (B)
- ✓ Port (C)
- ✓ Port (D)
- ✓ Port (E)

Port(A) = RA0 – RA4

Port (B) = RB0 – RB7

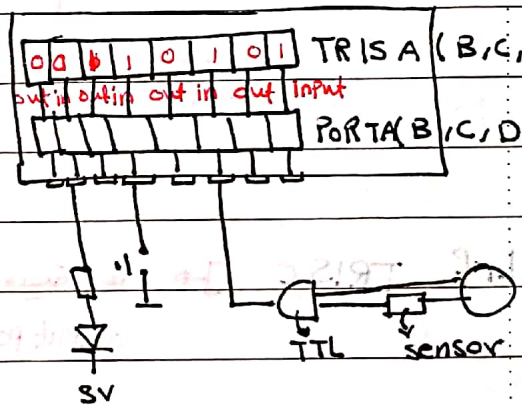
* طيب إزاي بربط الـ Pins مع

* عشان نتعامل مع الـ Pins اللي بربطها بيه الـ PIC مع البيئة الخارجية ، بستغل رجسترين اسمهم PORT / TRES Reg ، حيث كل Pin تبقى مربوطة مع input أو output element

مثلاً الـ Pin2 مربوط مع السينسور اللي بمثل input بالنسبة للمايكروكونترولر .

وكمان Pin4 مربوط مع الـ Push button اللي بمثل input بالنسبة للمايكروكونترولر .

والـ Pin6 مربوطة مع الـ LED اللي بمثل output للمايكروكونترولر .



* كل بورت المفروض يكون مربط مع الـ TRIS ، حيث انه الـ TRIS Reg بتتحكم بالداتا اللي جاية من الـ Port bits .

* لو كانت قيمة الـ bit2 من الـ TRISA تساوي 1 و الوالد تعني input لأنه شايل مع السينسور ، نفس الأمر الـ bit4 مربوط مع input قيمته 1 ، أما bit6 قيمتها 0 لأنه شايلة مع أوتبوت .

Subject Date No.

* input Port : هو البورت اللي بياخد داتا من ال outside world

* output Port : هو البورت اللي بيبعت داتا لـ "

* LEDs, 7 segment, motor, LCDs ⇒ OUTPUT. حالة ال write mode

* Switches, Push buttons, sensors, KeyPad, LCDs ⇒ INPUT حالة ال read mode

* MovLw 0X0F
MovwF TRISB

0000 1111 = 0F قيمة TRISB معناه انه

RB0, RB1, RB2, RB3 ← input Port كلهم لازم قيمتهم (1)

output Port كلهم

* clrf TRISC ← كل الـ TRISC قيمتهم صفر
output port معناه كله

* clrf TRISD ← output كله عبارة عن

comf TRISD ← يعني عملنا complement

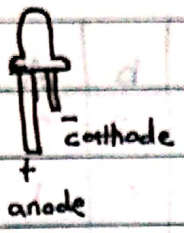
- input فبتصير قيمة 1111 1011 فكله بقى F F

* MovLw B'00110011'
MovwF TRISB
0,1,4,5 → input
2,3,6,7 → output

NOTEBOOK

Subject Date No.

* Light Emitting diodes LED :- هاد الـ LED واحد من الـ components اللي بنستعملها مع الـ PIC ، طبعاً التيار بيتنقل من الـ anode إلى الـ cathode .

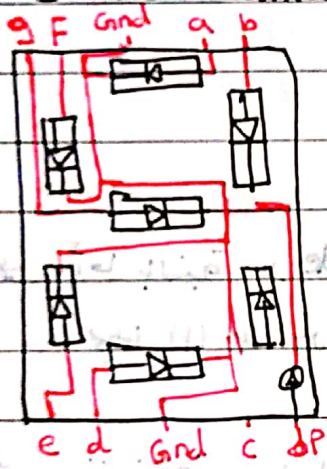


بنركب مقاومة مع الليد قيمتها بين 220Ω - 1KΩ ، بنسميها مقاومة Current Limiting Resistors.

* Seven Segment display :-

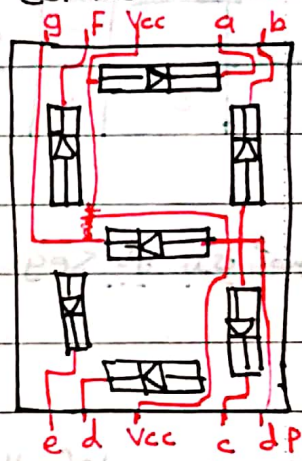
بالهاتوعين .

Common Cathode



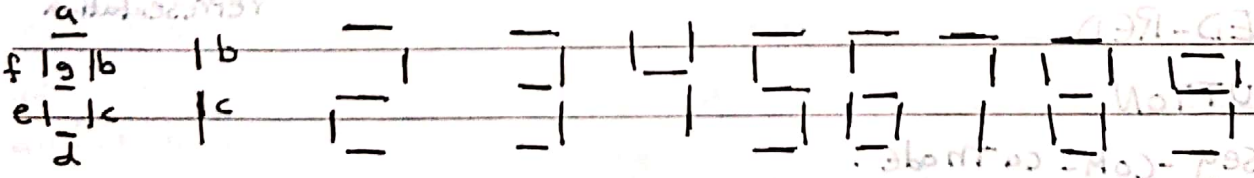
عشان تضوي أي segment فيه بنحط (0) .

Common anode



عشان تضوي أي segment فيه بنحط 5V .

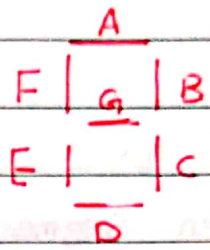
وبنحط current Limiting Resister مع كل segment .



Subject Date No.

* 7 segment display / common anode.

Digit	H	g	F	E	d	c	b	A	Hex value
0	0	1	0	0	0	0	0	0	
1	0	1	1	1	1	0	0	1	
2	0	1	0	0	1	0	0	0	
3	0	0	1	1	0	0	0	0	
4	0	0	0	1	1	0	0	1	
5	0	0	0	1	0	0	1	0	
6	0	0	0	0	0	0	1	0	
7	0	1	1	1	1	0	0	0	
8	0	0	0	0	0	0	0	0	
9	0	0	0	1	0	0	0	0	



* أما بالنسبة لـ cathode رح يكون نفس المبدأ بس الـ seg بتبقى بدي أضوي بطرح (1) من عندي.

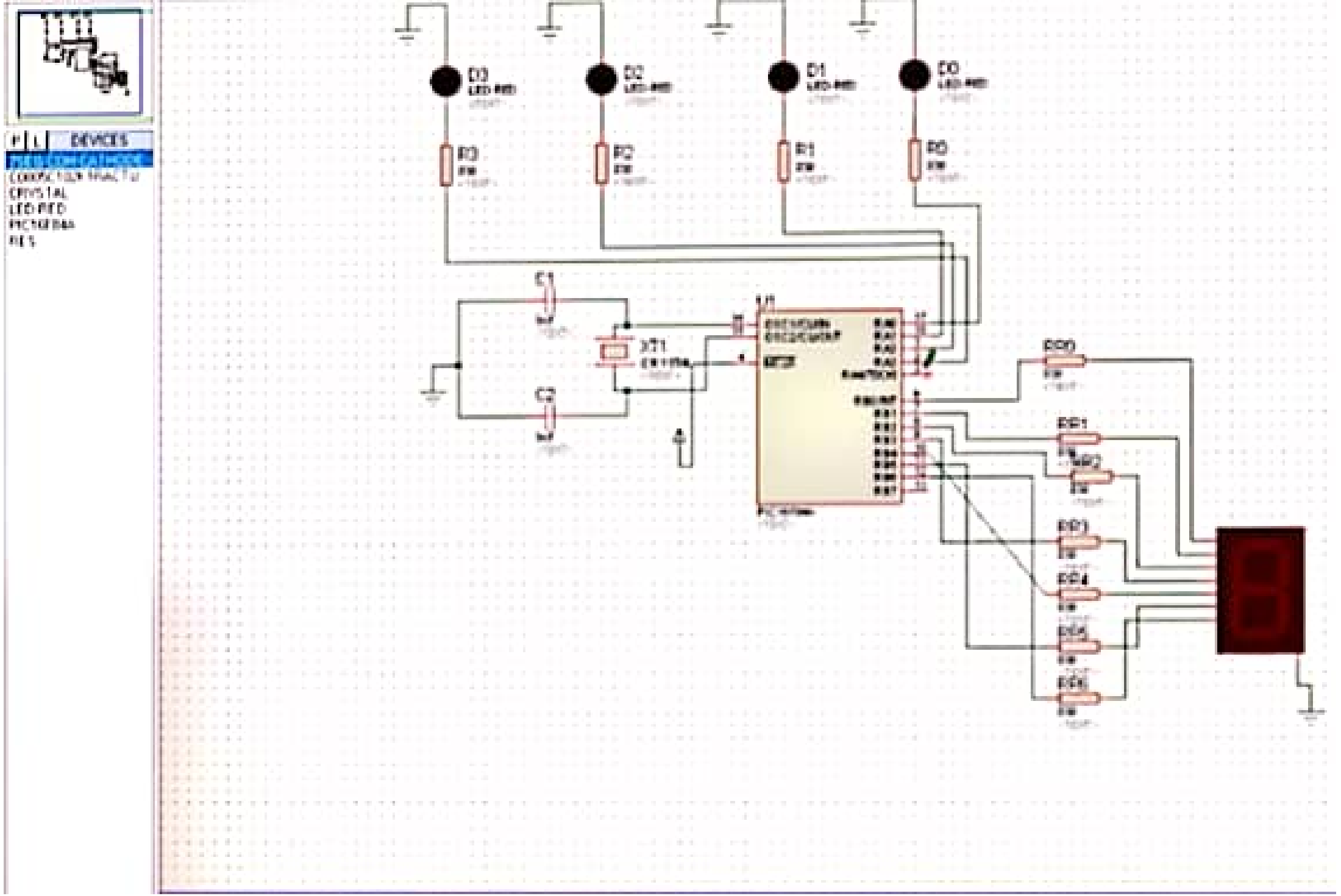
* بدنا نشتغل على البروتوس ، File ← new design ← default ← save as

* عشان نفعل 7 seg لازم نستعمل هدول الأشياء :-

- PIC 16F84A
- RES
- SW-SPDT
- LED-RED
- BUTTON
- 7seg-COM-cathode.
- CRYSTAL
- C0805C102K IRACTU

يحيث فيي كاونتر بعرض الارقام على 7 seg.

وال binary representation بعرضهم على الـ LEDS.


D3
D2
D1
D0
R3
R2
R1
R0
C1
C2
XT1
U1
DEVICES
CRYSTAL
LED-RED
RES

```
List     p=16F84a
     #include "P16F84A.INC"

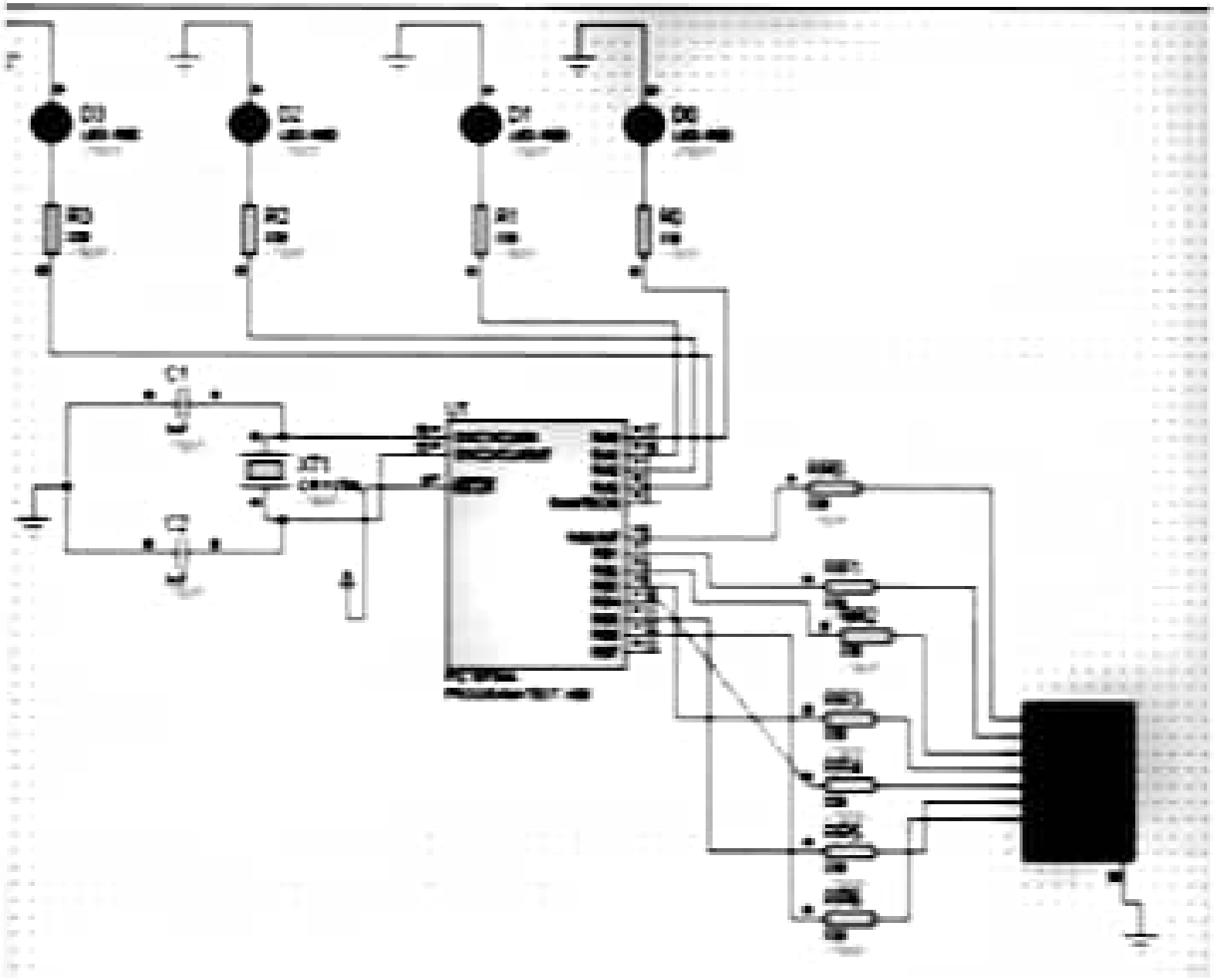
Zero          equ           B'00111111' ; 7-Segment Code for 0
One           equ           B'00000110' ; 7-Segment Code for 1
Two           equ           B'01011011' ; 7-Segment Code for 2
Three         equ           B'01001111' ; 7-Segment Code for 3
Four          equ      B'01100110'
Five          equ      B'01101101'
Six      equ      B'01111101'
Seven         equ      B'00000111'
Eight         equ      B'01111111'
Nine          equ      B'01101111'

;*****************************************************************
cblock 0x30
count
LED
RTC
C1
endc
     org 0x000
     bankse1 TRISA
     movlw 0x000
     movwf TRISA
     movwf TRISB
     bankse1 PORTA
     movwf PORTA
initial
     movlw 0x0a
     movwf C1
     movlw 0x0
     movwf count

display
```

CamScanner

to display the 6 number in the LED and the 7-segment



CamScanner

```
#INCLUDE "P16F84A.INC"
 CBLOCK 0X30
  V1
  V2
 ENDC
        ORG 0X000
        BANKSEL TRISA
        MOVLW 0X000
        MOVWF TRISA
        MOVWF TRISB
        BANKSEL PORTA
DISPLAY
        MOVLW 6
        MOVWF PORTA
        CALL DELAY
        MOVLW B'01111101'
        MOVWF PORTB
        CALL DELAY
        GOTO DISPLAY
DELAY
        MOVLW   0XFF
        MOVWF   V1
        MOVWF   V2
LOOP2
        DECFSZ  V1,     F
        GOTO    LOOP2
        DECFSZ  V2,     F
        GOTO    LOOP2
        RETURN
        END
```

CamScanner

Subject Date No.

هذا المثال عشان تعرض الارقام من صفر - 9

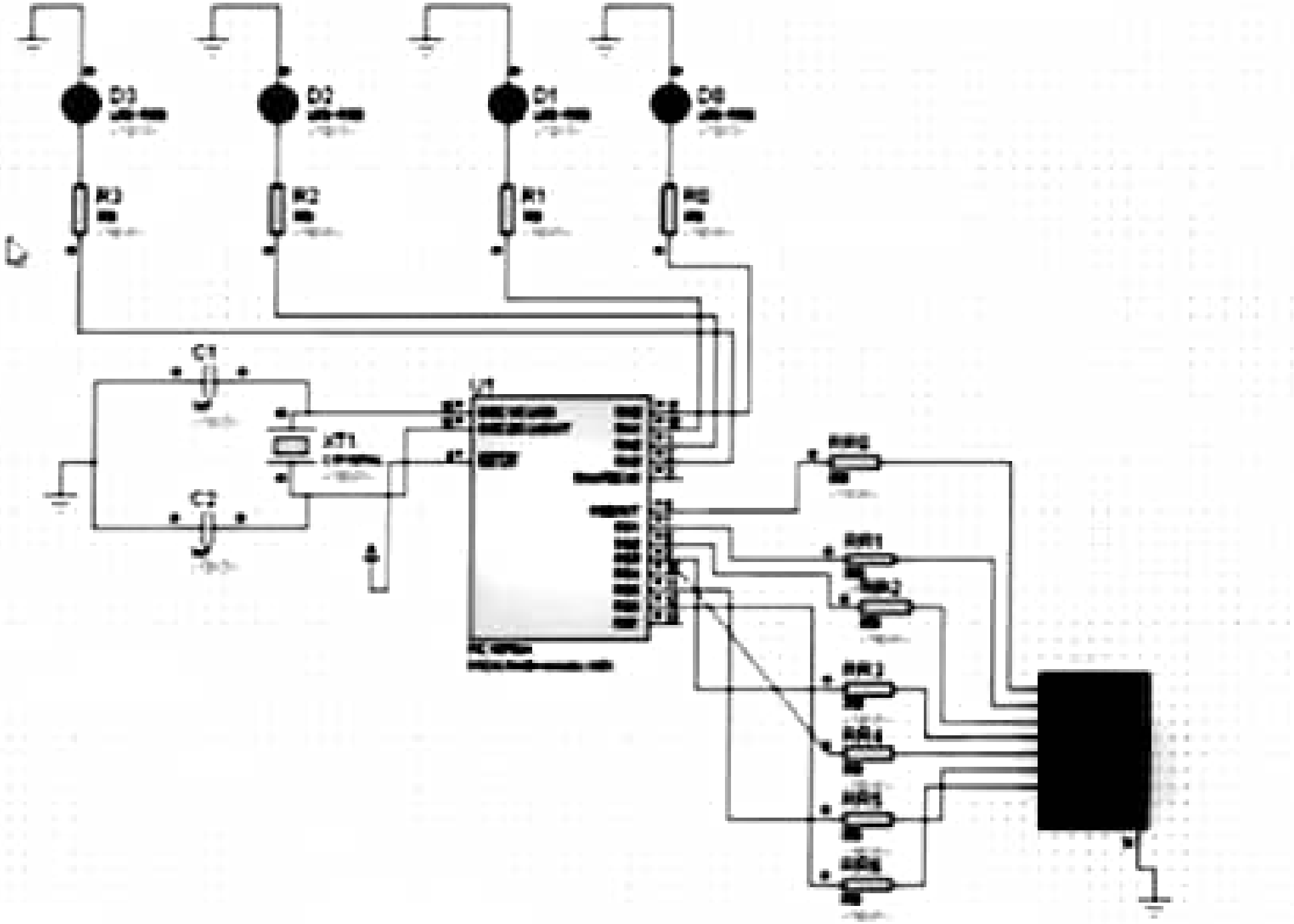
```
#include "P16F84A.Inc"
cblock 0x30
count
LSD
MSD
c1
endc
 org 0x000
banksel TRISA
movlw 0x000
movwf TRISA
 movwf TRISB
 banksel PORTA
  movwf PORTA
initial
   movlw 0x0a   -> لأنه توسع 10 أعداد من 0-9
   movwf c1
   movlw 0x0
   movwf count
display
 movf count,w
 call Update-Seven-Seg
movwf PORTB
call Simplest-Delay
movf count,w
movwf PORTA
call Simplest-delay
incf count,1
decfsz c1,1
```

```
Update-Seven-Segment
addwf PCL,F
RETLW B'00111111'
 //    B'00000110'
 //    B'01011011'
 //    B'01001111'
 //    B'01100110'
 //    B'01101101'
 //    B'01111101'
 //    B'00000111'
 //    B'01111111'
 //    B'01101111'
Simplest-Delay
MOVLW 0X255
MOVWF MSD
CLRF LSD

Loop2
DECFSZ LSD,F
GOTO Loop2
DECFSZ MSD,F
GOTO Loop2
RETURN
end
```

NOTEBOOK

CamScanner



CamScanner

PIC 16FXXXX Instruction Set:

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 1fff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1 (2)	00 1011 dfff ffff		1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1 (2)	00 1111 dfff ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 1fff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk		
RETFIE	-	Return from interrupt	2	00 0000 0000 1001		
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from Subroutine	2	00 0000 0000 1000		
SLEEP	-	Go into standby mode	1	00 0000 0110 0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z	

STATUS REGISTER

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC(1)	C(1)
bit 7							bit 0

INTCON REGISTER (ADDRESS 0Bh, 8Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

OPTION REGISTER (ADDRESS 81h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

20: spa	30: 0	40: @	50: P	60: `	70: p		
21: !	31: 1	41: A	51: Q	61: a	71: q		
22: "	32: 2	42: B	52: R	62: b	72: r		
23: #	33: 3	43: C	53: S	63: c	73: s		
24: $	34: 4	44: D	54: T	64: d	74: t		
25: %	35: 5	45: E	55: U	65: e	75: u		
26: &	36: 6	46: F	56: V	66: f	76: v		
27: '	37: 7	47: G	57: W	67: g	77: w		
28: (	38: 8	48: H	58: X	68: h	78: x		
29:)	39: 9	49: I	59: Y	69: i	79: y		
2A: *	3A: :	4A: J	5A: Z	6A: j	7A: z		
2B: +	3B: ;	4B: K	5B: [	6B: k	7B: {		
2C: ,	3C: <	4C: L	5C: \	6C: l	7C: |		
2D: -	3D: =	4D: M	5D:]	6D: m	7D: }		
2E: .	3E: >	4E: N	5E: ^	6E: n	7E: ~		
2F: /	3F: ?	4F: O	5F: _	6F: o	7F: ⌂		

LCD Commands:

Command	Binary D7	D6	D5	D4	D3	D2	D1	D0	Hex
Clear Display	0	0	0	0	0	0	0	1	01
Display & Cursor Home	0	0	0	0	0	0	1	x	02 or 03
Character Entry Mode	0	0	0	0	0	1	I/D	S	04 to 07
Display On/Off & Cursor	0	0	0	0	1	D	U	B	08 to 0F
Display/Cursor Shift	0	0	0	1	D/C	R/L	x	x	10 to 1F
Function Set	0	0	1	8/4	2/1	10/7	x	x	20 to 3F
Set CGRAM Address	0	1	A	A	A	A	A	A	40 to 7F
Set Display Address	1	A	A	A	A	A	A	A	80 to FF

I/D:	1=Increment*, 0=Decrement	R/L:	1=Right shift, 0=Left shift
S:	1=Display shift on, 0=Off*	8/4:	1=8-bit interface*, 0=4-bit interface
D:	1=Display on, 0=Off*	2/1:	1=2 line mode, 0=1 line mode*
U:	1=Cursor underline on, 0=Off*	10/7:	1=5x10 dot format, 0=5x7 dot format*
B:	1=Cursor blink on, 0=Off*		
D/C:	1=Display shift, 0=Cursor move	x = Don't care	* = Initialization settings

CamScanner

Subject Date No.

الكود صحيح وشغال ✓

:- اسئلة اضافية

Q: بدنا نعمل برنامج فيه Subroutine بجمع 4 أرقام وبطلع الأخير إلهم، ثم نقارن ناتج الجمع بالـ main. حيث اذا كان قيمة قردي بطبع حرف "O"، واذا كان زوجي بطبع حرف (E).

```
#include "P16F877a.inc"
cblock 0x30
num1
num2
num3
num4
result1
result2
endc
CALL AVG
btfsc result1,0
goto odd
goto even
odd
movlw A'O'
movwF result1
even
movLw A'E'
movwF result2
AVG
Movf num1,0
addwf num2,0
adwF num3,0
addwf num4,1
rrf num4,1
rrf num4,1
movf num4,0
movwf result1
return
end
```

Rotate right مرتين زي كأنه بقسم على 4

NOTEBOOK

Subject Date No.

Q ٤- بدنا نحل نفس المثال بكن باستخدام ال macro ؟

```
# include "P16F877a.inc"
Cblock 0x30
num1
Num2
num3
num4
result1
result2
endc
AVG macro
movf num1,0
addwf num2,0
addwf num3,0
addwF num4,1
rrf num4,1
rrf num4,1
movf num4,0
movwF result1
endm
AVG
btfsc result1,0
goto odd
goto even
odd
movLw A'0'
movwf result2
even
movLw A'E'
movwF result2
end
```


CamScanner

Subject Date No.

Q: بدنا نعمل برنامج بعمل implement لديكودر (3x8) ، اذا كان الانبوت B'000' فإن الاوتبوت B'0000 0001' ، واذا كان الانبوت B'011' فإن الاوتبوت B'0000 1000' وهكذا ---.

```
cblock  0x30
input
output
endc
                    movf  input,0
Call mylookup
add wf PCL,1        movwf output
retlw b'0000 0001'
retLw b'0000 0010'
retLw b'0000 0100'
retLw b'0000 1000'
retLw b'0001 0000'
retLw b'0010 0000'
retLw b'0100 0000'
retLw b'1000 0000'
movwf output
end
```


CamScanner

Subject Io interfacing Date No.

في هذا المثال، عنا كاونتر بعدّ 8 أرقام، بداية الكاونتر بحدده اليوزر من خلال الـ switch وبالضغط على الـ push button بنتقل من رقم إلى آخر ... كيف بنعمله ؟

* أول اشي بنشوف الـ PIC ، نحدد كل port هو input أو output ، بنلاحظ انه البورتات (RA0-RA4) مشبوكين مع الـ switch والـ push button ، معناته قيمة الـ Port A تحدد لازم TRIS A تكون IF عا انه كلهم input ، بهاي الحالة بنحدد كل input port .

```
MovLw 0F
MovwF TRISA ;   TRISA = 11111 / input
MovF PoRTA,0    read from the switch , now the value
                is save in w.
ANDLw 0F        بعمل and مع 0F لأنه بهمني أول أربعة bit من البورت A
movwF count     وبخزن النتائج في متغير اسمه count .
                هيك بكون قرأنا القيمة من الـ switch وخزناها في الكاونتر .
                وهاد كله اسمه [read from the environment] .

CLRF TRIS B     هون بدنا نكتب على 7segment أو نعرض الدوبتون من خلال
                الـ led يعني [display output] نعرض الدوبتون .
                وهاد بصير من خلال البورت B ، يلي بتكون قيمته كلها صفر
                لأنه TRIS B = 0 output port .

MovwF PoRTB     بنقدر نكتب على أي بورت من خلال movwF . حيث تجمع القيمة
                المخزنة في W إلى البورت المعني .
```

NOTEBOOK

Subject Date No.

بما انه البورت B شابك مع الـ 7 segment مشان لازم يكون عنا lookup table عشان كل الأرقام من (0-F) ، ثم ترجع القيمة يلي تمثل الرقم إلى اللاتش بتنقل لاحقاً إلى PORT B.

من خلال الـ push bottun، بدك تنتبه انه الانتقال بيه رقم وآخر يكون من خلال الضغط على الـ push bottun يلي مشبوك على الـ Vcc من طرف وعلى الـ ground من الطرف الآخر.

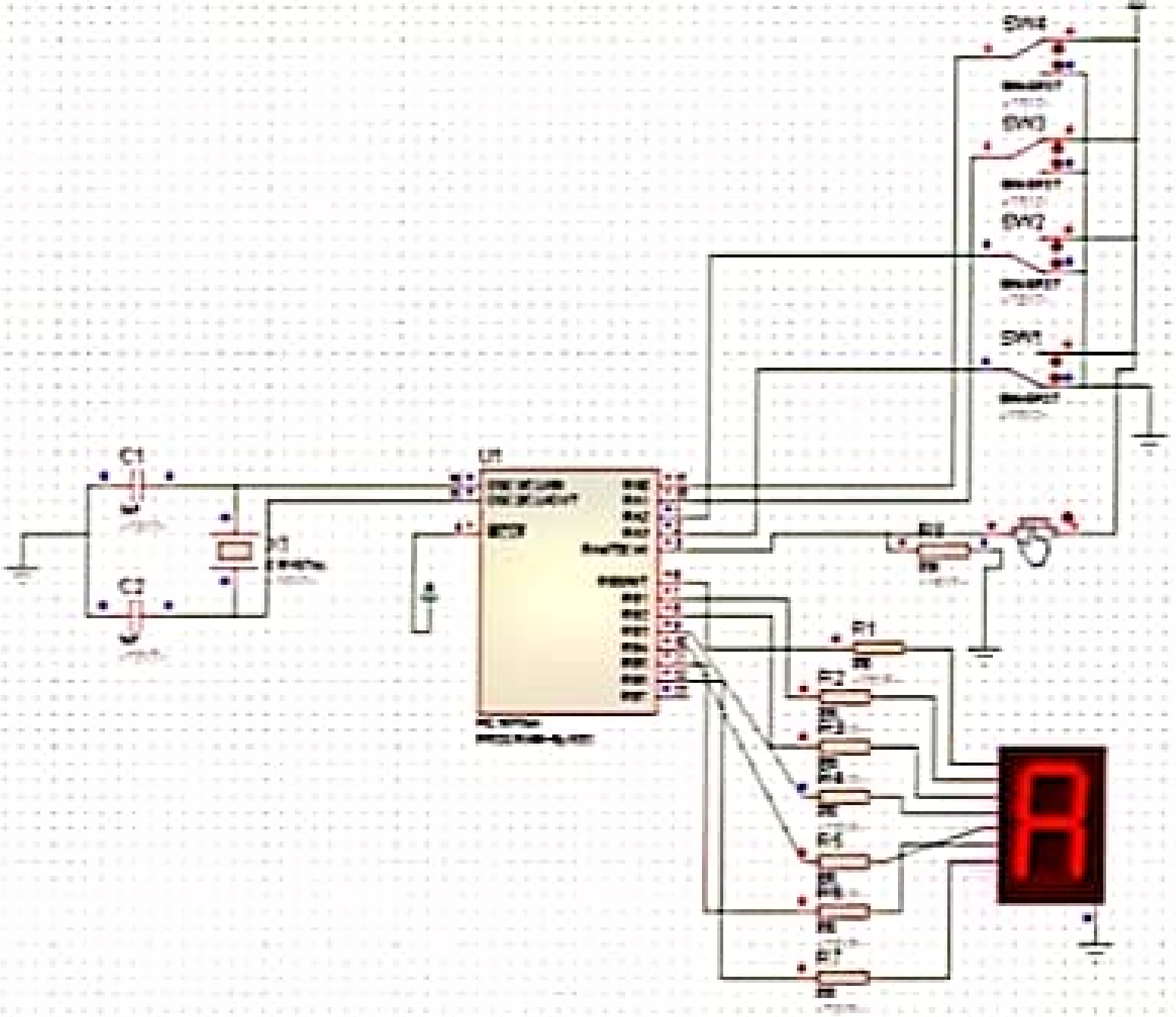
ولكن تلقائياً القيمة يلي بتدخل على RA4 دائماً (0) فبالتالي يلي بتكبس فيها على الـ push bottun رح تتغير قيمة RA4 من صفر إلى واحد.. عشان نطبق هاي النقطة بنستعمل هالأوامر:-

```
btfss PORTA,4
goto agin
```

لأنه بالكفة اللي بتكون فيها A4 تساوي (1) بتقفز وتطلع من الـ waiting، وطول ماضي تساوي صفر فبتضل بالـ infint loop

- Posh bottun ويعمل Pooling وينتظر كبسة من الـ

* الكود كامل مكتوب في الصفحة التالية:-



CamScanner

Subject Date No.

```
Include " P16F84A.inc"
cblock 26
LSD
MSD
count
X
Endc
org 0x00
Banksel TRIS A        ] تعريف PORT A كأنبوت
movlw 0x1F            ]
movwf TRIS A
CLRF TRIS B           ] تعريف PORT B كأنبوت
BANKSEL PORTA
L1
movlw 8               ] عبارة عن الكاونتر رح يعد 8 أرقام فبخلط رقم 8 في X.
MovwF X               ]
movF PORTA , W        ]
ANDLW 0F              ] بنقرأ القيم من السويتش المربوط على PORTA وأخزنه في count.
movwF count           ]
main
CALL DECODE           ] بنروح على look up table اللي بمثل الأرقام من 0-F وبرجع وتمثيلها على w عشان 7seg
movwF PORTB           ] رح يعرض النتيجة على PORTB
Call delay
Again
```

```
btFSS PORTA , 4       ] صوت عشان كبسة ال push bottun
Goto again            ]
INCF count , 1        يلي حتى صناها
movF count , 0        حال كبسنا عليها.
DECFSZ x , 1          بنزيد قيمة الكاونتر
goto main             بمقدار واحد و بخزن قيمة
goto L1               الكاونتر في (W) وبعدها بقفز
DECODE                رقمين بنقص الـ X بمقدار (1)
addwF PCL , F         فبضيف 7 ، مادام ماوصلت
RETLW B'00111111'     X لصفر فبنزفل بهاد
RETLW B'00000110'     اللوب بعرض الأرقام
//  B'01011011'       وفي الكلمة اللي
//  B'01001111'       بتصير فيها قيمة
//  B'01100110'       X=0 بنطلع
//  B'01101101'       من اللوب
//  B'01111101'       وبنعمل skip
//  B'00000111'       وبعمل goto L1
//  B'01111111'       وبيشي الكاونتر
//  B'01101111'       من البداية.
//  B'01110111'
//  B'01111100'
delay
MovLW 0X255
MovwF MSD
CLRF LSD
loop2
DECFSZ LSD, F
GOTO LOOP2
DECFSZ MSD, F
GOTO LOOP2
RETURN
end
```

CamScanner

Subject ex (5) :- interrupt . Date No.

* عندي طريقتين للتواصل بين المايكروكنترولر والاكسترنال سيستم :-

[1] Polling : السيستم بنفذ الأمر ، زي ما شرحناها بالتجربة السابقة لما استخدمنا الـ push button لما يوصله كبسة من الـ push button ، وطبعاً هاد الوقت ضائع [وقت انتظار الكبسة] لأنه خلاله ما بعمل المايكرو كونترولر أي اشي.

[2] interrupt : بكون البرنامج بنفذ بأمر معين ، وفجأة بجيه أمر خارجي فبوقف شغل عن الأمر السابقة ويبدأ بتنفيذ الأمر الخارجي ، وعن خلال الـ interrupt بستجيب المايكروكنترولر لأي event في أي لحظة بغض النظر عن الأمر اللي بنفذه حالياً.

* الـ interrupt بغير الـ Program flow وبشتغل لتنفيذ الـ interrupt service routine

interrupt is an event that causes the microcontroller to halt the normal flow of the Programe and execute anther Program called the interrupt service routine.

* interrupts sources can be external and enternal.

* there are 2 types of interrupt : maskable and non-maskable

maskable: can be enabled/disabled by setting/clearing some bits.

non-maskable: cant be disabled and they alweyes interupt the CPU.

يعني أنت بتقدر تتجاوزها أو بتطفيها [بتقدر تتجاوزها].

هون اذا احنا اتربت مسجل نقدر نتجاوزها ونكمل الـ CPU عمله الحالات وما بنقدر نطفيها.

NOTEBOOK

Subject Date No.

ملخص سريع للـ interrupt :-

* يكون المايكروبروسيسر بنفذ برنامج وفجأة بيجيه interrupt فبنقطع عن تنفيذ البرنامج الأصلي وبروح ينفذ الـ interrupt ، وبس يكمله برجع للبرنامج الأصلي .... بأي الطريقة احنا كنا الوقت بيعدي المعدود في الـ (polling) حلينا شكلة

* طيب لما نروح ننفذ الـ interrupt وين بالضبط ؟ دائماً الانتربت بكون مكتوب على الأدريس 4 بروح بجاله fetch من هناك بواسطة org 0x004 ... وبلش أنفذ الـ instructions بالترتيب لحد ما أوصل لعند RETEF هيك بفهم انه الانتربت خلص ولازم أرجع للبرنامج الأصلي .

آدريس

* بالنسبة للانتربت لما أروح أنفذه ، فهو عبارة عن مجموعة من الـ instructions المتتالية يعني زي الـ subroutine ففيسميه Subroutine service .

* الـ interrupt إله مصدرين : إما مصدر خارجي زي لما يجيني إشعار من حساس الحرارة أو داخلي يعني من داخل سيستم الـ pic زي لما الكاونتر يكون بعد لحد 255 فبس يوصل 256 يعمل [timer overflow interrupt] و فبروح ببعتة signal interrupt من الـ pic نفسها عشان يوقف اشي معين أو يشغل كود معين .

Subject Date No.

* each interrupt has a flag (a bit) that is set ~~whether~~ whenever the interrupt occurs.

* non maskable interrupts in 16F84A.] *

* Sources of the interrupt * :-

1. external interrupt. [هادا اهم نقطة]
   - the only external interrupt input.
   - the input is multiplexed with RB0 pin of port B.
   - [it is edge triggered.] هاد الـ bit بيجي من عند الانتربت

   multiplexing فهو مجموع RB0/INT
   فيا يكون interr.. أو يكون RB0
   [يتفعل لغرضين].

2. Timer overflow interrupt.
   - it is an internal interrupt that occurs when the 8-bit timer overflows.

3. Port B interrupt chang.
   - an interrupt occurs when a change is detected on any of the upper 4 bits of port B.

4. EEPROM write complete interrupt.

CamScanner

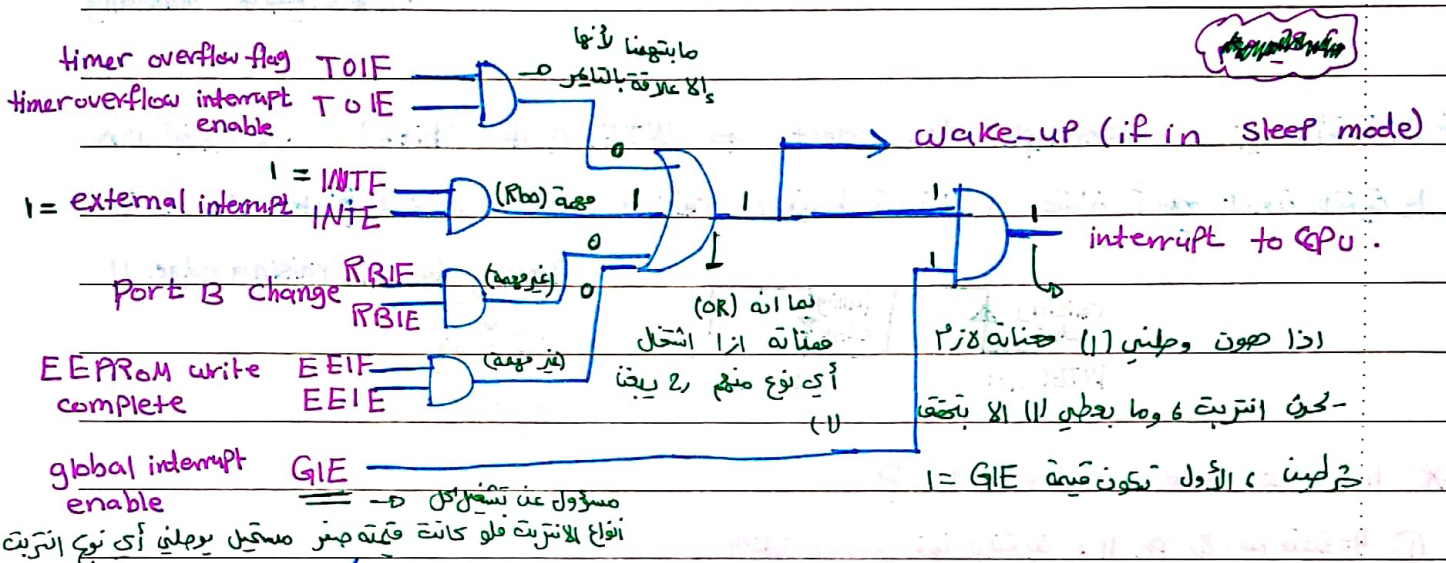
* بنفهم من الرسمة انه عشان يصير interrupt لازم كل من GIE و INTE و INTF تكون قيمتهم 1 بتشغل اذا اجا high voltage

Subject Date No.

1 ، غير هيك بصير الانتربت .

* كل interrupt إله flag و enable ، حيث أنه القارئ بتبين اذا صار interrupt أو لا .

* عشان الـ PIC تسمح باستقبال أي interrupt لازم نفعل الـ global interrupt enable (GIE)



* في عنا ريجسترين بتتعامل معهم لما نكتب interrupt وهمة عبارة عن readable و writable Registers .

INTCON Reg / OPTION Reg

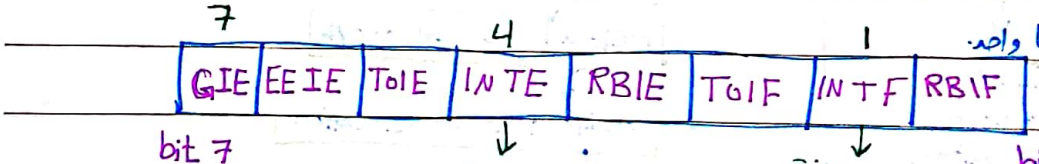
INTCON Reg: يحتوي على مجموعة من Various enable bits for all interrupt sources

OPTION Reg: يحتوي على control bit تهيئة لاستخدام external interrupt

* بهاد اللاب رح نركز على الـ external interrupt وهو الإنتربت اللي بستقبله الـ PIC عن طريق RB0/INT

* بهاد المثال ، نلاحظ انه القيمة اللي بتدخل على الـ pin هي صفر ، فلما نكبس على Push button رح تتغير من صفر لواحد ، وهاد الانتربت بشتغل على الـ rising edge .

* الـ PIC ما بتستقبل أي interrupt عشان تفعل الانتربت بنشوف الـ Intcon Reg فيها الـ bit7 (GIE) بخلي قيمتها واحد.

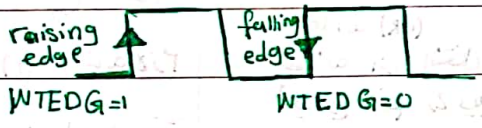


NOTEBOOK

Subject Date No.

وبفعّل INTE ، INTF بتغير قيمتها ! لما يحدث ال interrupt sur... بترجع قيمتها صفر [بس لازم نعدل قيمتها manually ما بتغير لحالها].

هاد بتروح على ال (bit 6) ⇐ INTEDG ⇐ interrupt edge select ، زي ما أخذنا قبل شوي ! عشان أسمح لحدوث الانترببت على raising edge ، فبحط قيمتها 1 ، لانه ال interrupt يحدث على ال falling edge ، أو raising edge



بخلي قيمتها صفر

* how to use interrupts ?

① 4=بغير قيمة PC ، الخطوة اللي بصير فيها الإنترببت ، ال PC بنقل إلى لوكيشن 4 وبنفذه.

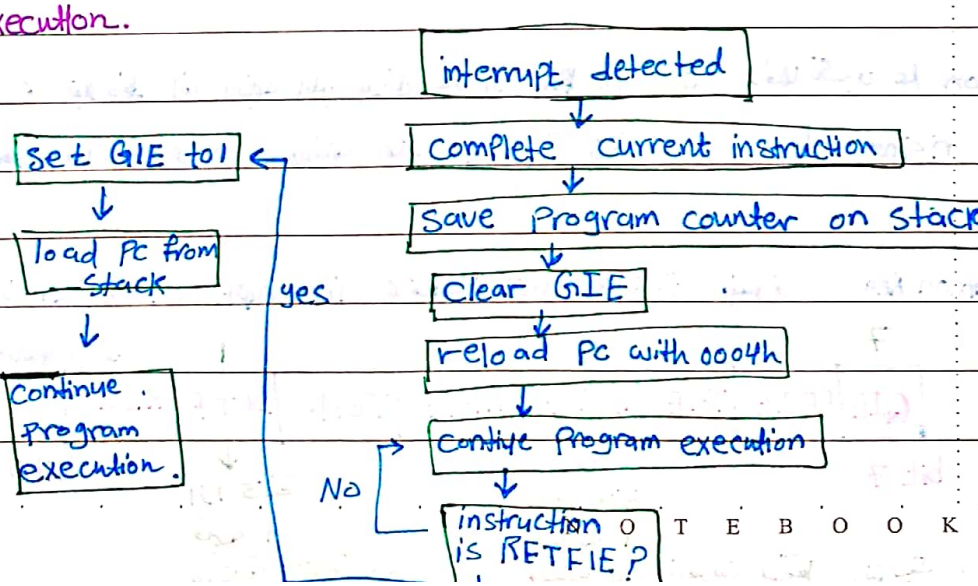
وهاد معناه انه ال PIC بنقل إلى لوكيشن

② Clear the flag of used interrupt in the INTCON register.

③ enable the corresponding interrupt by setting its bit in INTCON Reg.

④ enable global interrupt by setting the GIE bit.

⑤ end the interrupt subroutine with RETFIE instruction to resume program execution.



CamScanner

Subject Date No.

```
# include "P16F84A.INC"
cblock  0X20
MSD
LSD
TEMP1
TEMP2
endc
org 0X00
GoTo main
org 0X0004
GoTo INT_SVC
```

بدنا نكتب كود يضوي الليد على شكل (1010 ، 0101)
ومجرد ما نكبس الـ Push.. بصير يضوي على شكل (1100 ، 0011)

→ يروح على الـ main وفي أي حالة
بصير انتربت ، يروح على لوكيشن ٤ ويشوف الأمر
interrupt service routine.

```
main
BANKSEL  INTCON
BCF      INTCON , INTE
BSF      INTCON , INTE
bsf      OPTON_REG , INTEDG
BsF      INTCON , GIE
```

INTCON فيه ٢ بيتس لازم أدخل
INTE قيمته صفر و INTE قيمته ١
لانه لسا ببداية البرنامج فلما يصير انتربت بصير!

raising edge بعمللها enable اذا كان الانتربت على
وصفر اذا كان على الـ ~~falling~~ edge.
falling enable بعمللهم

```
banksel TRISA
movLw 0
movwF TRISA
movLW 1
MovwF TRISB
```

بما انه PortA مربوط معه الليد معناته قيمته صفر
وبما انه PortB مربوط معه الكبسة معناته قيمته ١

```
~~banksel~~
banksel PORTA
MovLw B'0101'
movwF PORTA
CALL delay
```

NOTEBOOK

Subject Date No.

```
movLw B'1010'
movwF PORTA
CALL delay
GoTo main
INT - SVC          ] هاي بداية الانتربت
movwF TEMP1        ] حفظنا قيمة w برجيستر عشان ما تضيع
BANKSEL INCON
bcf INCON , GIE    ] GIE لازم نعمل clear
banksel portA
movLw B'1100'
movwF PortA
CALL delay
CALL delay
movLw B'0011'
movwF PortA
CALL delay
CALL delay
BANKSEL INTCON
BcF INTCON , INTF  ] clear نعمل للفلاغ
BSF INTCON , GIE   ] GIE نعمل set
BANKSEL PORTA
movF TEMP1,0       ] restore w
RETFIE             ] انهاء الinterrupt
delay
movLw 0XFF
movwF MSD
clRF LSD
loop2
DECFSZ LSD,F
GoTo loop2
DECFSZ MSD,F
GoTo loop2
RETURN
end
```

عشان نقدر ننفذ الانتربت
مش واحنا بننفذ بالانتربت
تلاقي كمان انتربت جاي ننفذه
الوقت ... بس نكمل شغل برجع
بنفسويه .

Code
بتبع
interrupt

بس نكمل كود الانتربت بدنا نرجع نفعّل صعل

عشان ما يرجع
للانتربت مرة تانية

لأنه بدنا نرجع
لـ main

NOTEBOOK

Subject Date No.

* اثناء تنفيذ الـ interrupt ممكن تتغير قيمة الـ working register فمشان نحفظ قيمتها بنخزنها في TEMP مثلاً. ⇐ MovwF TEMP ،

وبعد ما نخلص الـ interrupt ، المفروض نرجع قيمة W ⇐ MovF TEMP, W

* كيف نحدد الـ Sourse of interrupt ؟

من خلال الـ flags ، فكل من RBIF / T0IF/INTF اذا كانت قيمتهم (1) معناته هاد هو مصدر الانتربت.

فرح نستخدم هاد الانتركشن حتى نفحص قيمتهم ⇐ btfsc INTCON ,0

هاد الانتركشن رح يفحص قيمة (RBIF) اذا كانت قيمته صفر أو واحد.

واذا كان مش صفر ، برجع بفحص اللي بعده ⇐ btfsc portb-int

ثم بشيل على اللي بعده ، وبشيك على INTF ⇐ btfsc INTCON ,1

⇐ external in--- goto ext-int

⇐ T0IF " " " " btfsc INTCON ,2

⇐ timer overflow routine goto timer-int

⇐ test EEPROM write complete flag btfsc EECON1 ,4

⇐ EEPROM write complete routing goto eeprom-int

CamScanner

N_3	N_2	N_1	N_0
0	0	0	0
0	0	0	1

Subject Date No.

* برنا نكتب كود بقرأ القيم من الـ switches على البورت A ، بودّيها على w ، وبعتها لـ look up table تبعت الـ 7-segment ، عشان يعرض الرقم عليها بناءً على قيمة الـ switches.

* شبكنا الـ Push button على Port B ، بحيث لما أكبس عليها فإن الرقم المعروض على الـ seven segment رح يبلش يعد تنازلي .... وطول ما هو بعد تنازلي هشو ما غيرنا بالسويتشات ما رح يفرقهم ... لحد ما يكمل العد التنازلي.

كود الـ main :

```
BANKSEL TRISA
movLw   0FFh        ← لأنه عنا 4 switches مشبوكين على A عشان يضووا لازم تكون F = 1111 فيفهم
movwF   TRISA

movLw   0000 0001   ← هاد(1) بقابل الـ Push button
movwF   TRISB       ← شايفين على الـ 7seg والـ (0) للـ 7seg.

BANKSEL PortB
movF    PortA , w   ← شايفين على الـ 7seg (outPut)
movwF   Counter     ← عبارة عن متغير برنا نخزن فيه القيمة يلي بنقرأها من السويتشات عشان لما يصير الانتربت بيلش يعد تنازلي من عندها.
CALL    Seven segment ← بنروح على الـ lookup table تبع الـ 7seg عشان نرضي عليه النتايج.
movwF   PortB       ← ودينا النتيجة على PortB عشان يعرض النتيجة.
GoTo    main        ← عشان نضل نتكرر هاي العملية.
```

NOTEBOOK

CamScanner

Subject Date No.

كود الـ interrupt

```
:
:
Banksel  PortB
L1  movF   Counter , W   → بدنا نقرأ القيمة اللي خزناها بالكاونتر عشان نقدر
    CALL   seven segment   تنازلي من عندها.
    movwF  Portb   → 7 seg هون بدنا نعرضها على
    delay   → عشان نعرض النتيجة
    GOTO L1   → احنا بدنا نضل نقرأ من الكاونتر ونعرض
    BSF     النتيجة ، فبدنا لون نكن في خرج
    DECFSZ  Counter , F   انه لازم أنقص واحد (عد تنازلي) وبس
    GOTO L1   أول الصفر خلص بطلع من الانتربت.
    movLw 0
    CALL seven seg   → عملنا هيك لأنه فوق لما يوصل الكاونتر
    movwF Portb   ا ، رح يطلع من الانتربت ، مش لما يوصل
    CALL delay   صفر ، فلما يعرض الصفر فعشان أعرضه
                 كتشبه بكود الحال.
```


CamScanner

Subject exp(6): controlling the LCD Date No.

* من خلال الـ LCD نقدر نعرض أرقام وحروف، الـ LCD بتتحكم فيها الـ chip اللي اسمها HD 44780.

* الـ LCD فيها أحجام وأشكال مختلفة بتتراوح بين 8, 16, 20, 24, 32, 40 Characters.

standard in 1, 2 and 4 Line versions.

* بهاي التجربة رح نتعامل مع LCD - 16 Char - 2 Lines رقمها (LM016L).

1	2	3	4	5	----	39	40
00	01	02	03	04	....	26	27
40	41	42	43	44	....	66	67

* موظم الـ LCD فيها 14 Pin
- D0-D7
- 3 control line
- 3 Power Line

* عشان تشتغل الـ LCD، لازم تكون (VDD=5V)، (Vee=0)، Vss بتتحكم

بسطوع الـ LCD فعشان تكون السطوع أكبر ما يمكن، لازم تكون قيمتها صفر.

* اذا كانت الـ LCD فيها (16 Pin)، فال 2 Pin الزيادة بيستخدموا لأغراض الـ back lines.

PIN no	Name	function
L+	anode	back ground light
L-	Cathode	//
1	Vss	ground
2	vdd	+ve supply
3	vee	contrast
4	RS	register Select
5	R/W	read / write
6	E	enable
7	D0	data bit 0
8	D1	// 1
9	D2	// 2
10	D3	// 3
11	D4	// 4
12	D5	// 5
13	D6	// 6
14	D7	// 7

CamScanner

Subject Date No.

* Reg/sel :- هي Register select بتتحكم ال LCD من خلالها بالداتا اللي واصلينها هل هي جاية من Command أو من Character .

* R/w :- استخدام ال LCD للقراءة والكتابة ، احنا باللاب بنا نستعملها للكتابة فداءاً قيمتها 0 .

* عشان نبعت كوماند أو داتا إلى LCD ، المفروض نكتب ال ascii code اللي على اللابن من D0-D7 .

* اذا كانت قيمة Rs واحد ، ع تفهم ال LCD انه هاي القيمة اللي واصلتها هي ascii value ، أما اذا كانت قيمة Rs صفر ، معناته ال ascii يلي وصل ال LCD هو كوماند .

* عشان يتم تنفيذ الانستركشن ، لازم تكون قيمة R/w=0 ، وكمان لازم يصير Pulse ع الـ Enable من صفر إلى واحد .

* Steps to send character to LCD :-

1) Place the ascii character on the D0-D7 Lines.
2) register select (Rs) =1 to send characters.
3) "enable" Pulse (set hight - delay - set low).
4) delay to give LCD the time needed to display the characters.

```
Send-char
movwF PortC
bsf PortB,2
bsf portB,3
nop
bcF PortB,3
call delay
return
movL W,(X)
call send-char
```

* أي Value بيتم عرضها على location بيتم الوكيشن فارغ . طالما بيتم بيتم الوكيشن فارغ .

Subject Date No.

* اذا بدنا نبعت كوماند ، بتبع نفس الخطوات بس الفرق انه RS=0 وبعين الكوماند على PortB

* LCD commands :-

(1) clear display

```
movLw  0x01
call   send-cmd
```

(2) display and cursor home

```
movlw 0x02
call send-cmd
```

(3) display ON/off cursor (الرجوع إلى الشيت).

```
movLw 0x08
call send-cmd
```

* مثلاً بدنا نكتب حرف على وسط الـ LCD بالسطر الأول.
Location 6

```
movLw 0x86
call send-cmd
```

0x86 → عشان أحرك الكيرسر لهاد المكان.

1 0000 110
D7 رقم 6

لو بدنا نكتب بأول Position بالسطر الثاني :-

40h = 1 00 0000
D7 40h

```
movLw 0xC0
call - send -cmd
```


CamScanner

Subject Date No.

* قبل ما أبلش استخدم الـ LCD لازم أعملها (initalize) :-

movLw 0x38
call send-cmd → 8-bit mode , 2-line display , 5x7 dot format

Initial

Banksel TRISC
clrf TRISC
CLrf TRISB
Banksel PORTC

→ لحدد البورتات اللي تشتغليه عليهم ، هيه الثابتة على البورت وهيه ثابتة على أوبشون (حسب برمجة).

movLw 0X38
Call send-cmD → LCD الـ initiize صوت بتعمل

movLw 0x0e → display cursor , curser underline on , Blink off
Call send-cmd

movLw 0x02 → display cursor home (أرجع الكيرسر للوكيشن صفر)
call send-cmd

movLw 0x01 → clear display.
Call send-cmd

CamScanner

Subject Date No.

```
main
movLw 'A'
call  send-char
movLw 'B'
call send-char
movLw 0XC0
call  send-cmd
movLw 'C'
call  send-char
movLw 'D'
call  send-char
goto  FIN

send char
movwF  PortC
bsF  PortB, 2
bsF  PortB, 3
nop
bcF  PortB, 3
call  delay
return
```

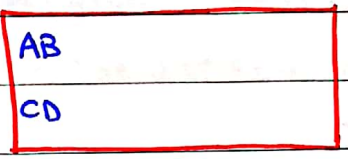
كود ثابت

CamScanner

Subject Date No.

```
send_cmd
    movwF   portC
    bCF   PortB, 2
    bsF   Port B, 3
    nop
    bcF   PortB, 3
    call  delay
    return

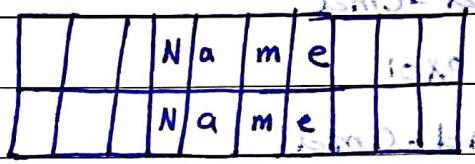
delay
    ;
```



CamScanner

Subject Date No.

write a program to display your name in two lines on the LCD as shown in Figure :- سؤال إضافي :-



initilize the LCD as the following conditions : \ 8bit mod, 2 line display, 5x7 dot format.

\ display on, cursor on, blink off.

\ display and curser home.

\ clear display

الحل :- ريكورد جميع وشغال

```
#include "P16F877a.inc"
cBlock 0x20
V1
V2
endc

Banksel TRISC
CLRF TRISC
CLRF TRISB
Banksel PORTC

movLw 0x38
CALL SEND-CMD
movLw 0x0e
Call SEND-CMD
```


CamScanner

Subject Date No.

```
movLw : 0X02
call send-cmd
movLw 0X01
call send-cmd

main
movLw 0X83
call send-cmD
movLw 'N'
call send-char
```

نرسل الحروف ثم نستدعي
send-char
(تأكد من الكتابة)

```
movLw 0XC3
call send-cmD
movLw 'M'
call send-char
```

تأكد نفس الخطوة لإرسال
حروف الاسم.

```
Goto Finish


seND-char
movWF PortC
BsF PortB, 2
"   "  , 3
Nop
BCF PortB, 3
call delay
return

Send-cmd
movWF PortC
BCF port B, 2
BSF port B, 3
Nop
BCF Port B, 3
call delay
return

delay
movLw 0XFF
movWF V1
movWF V2
loop
DECFSZ V1, 1
goto loop
DECFSZ V2, 1
goto loop
return
Finish
end
end
```

NOTEBOOK

LCD command control codes

Command	Binary D7	D6	D5	D4	D3	D2	D1	D0	Hex
Clear Display	0	0	0	0	0	0	0	1	01
Display & Cursor Home	0	0	0	0	0	0	1	x	02 or 03
Character Entry Mode	0	0	0	0	0	1	1/D	S	04 to 07
Display On/Off & Cursor	0	0	0	0	1	D	U	B	08 to 0F
Display/Cursor Shift	0	0	0	1	D/C	R/L	x	x	10 to 1F
Function Set	0	0	1	8/4	2/1	10/7	x	x	20 to 3F
Set CGRAM Address	0	1	A	A	A	A	A	A	40 to 7F
Set Display Address	1	A	A	A	A	A	A	A	80 to FF

1/D:	1=Increment*, 0=Decrement	R/L:	1=Right shift, 0=Left shift
S:	1=Display shift on, 0=Off*	8/4:	1=8-bit interface*, 0=4-bit interface
D:	1=Display on, 0=Off*	2/1:	1=2 line mode, 0=1 line mode*
U:	1=Cursor underline on, 0=Off*	10/7:	1=5x10 dot format, 0=5x7 dot format*
B:	1=Cursor blink on, 0=Off*		
D/C:	1=Display shift, 0=Cursor move	x = Don't care	* = Initialization settings

CamScanner

Subject Exp(7): Arduine and sensors Date No.

* بهاي التجربة رح نتعلم كيف نوصل الأردوينو مع السينسورات ، وكيف نعاملها interface مع الأردوينو.
* احنا بنعرف انه أي امبيديد سيستم بحتاج ياخد قراءات وداتا من العالم الخارجي ويعمل عليها controlling ، هاي القراءات بتجيبه عن طريق ال sensors وال actuators ... بتتحكم بال Physical Environment.

* Sensor : is an electronic device that is constantly measuring a physic variable, like temperuture, distance, ... and then transforms the phyical variable in an electric signal.

* actuator: is an electronic device that transforms and electric signal in a physic variable (light, sound ...).

* there are 2 types of sensors :- digital and analog sensors.

digital	analog
only detects 2 possible statuse (0, 1).	measures continuous the variable and detects any proportional value between 0-1.

* بهاي التجربة رح نتعامل بس مع ال digital sensors لأنه تشغيلا رح يكون سهل Simulation.

Subject Date No.

[1] Flame sensor :- هاد السينسور بحدد اذا في لهب أو لا ... بكون حساس لهيك ... wavelength = 700 - 1100 nm

digital output ، analog output ... بكون في threshold معينة

real time output
Voltage signal .

اذا وصلها بتحكي انه في لهب

واذا ما وصلها ما بتحكي أي اشارة

عليه مجموعة من الـ pins :-

GND / D_o / A_o / Vcc

D_o ← digital output

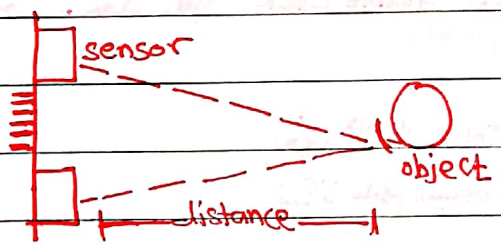
A_o ← analog output

Vcc ← 5v → analog
3.3v → digital

[2] smoke sensor :- حساس للغازات ، ونسبة الـ sensitivity بتختلف باختلاف نوع الغاز

بتحكم بالـ sensetivity من خلال الـ potentiometer .

[3] ultrasonic sensor :- مبدأ عمله انه بيبعث موجات فوق الصوتية تردداتها 40000H بتنتقل بنفس سرعة الصوت ، هاي الموجات اذا صادفت object بالطريق رح ترتد إلى السينسور ، فبعدين بنقدر نحدد اذا في object وبحدد المسافة بينه وبين السينسور .



عليه مجموعة من الـ pins :-

GND / Trig / echo / Vcc

distance = time * speed of sound in air (343 m/s)
2 (المسافة بين السينسور والاوبجيكت)

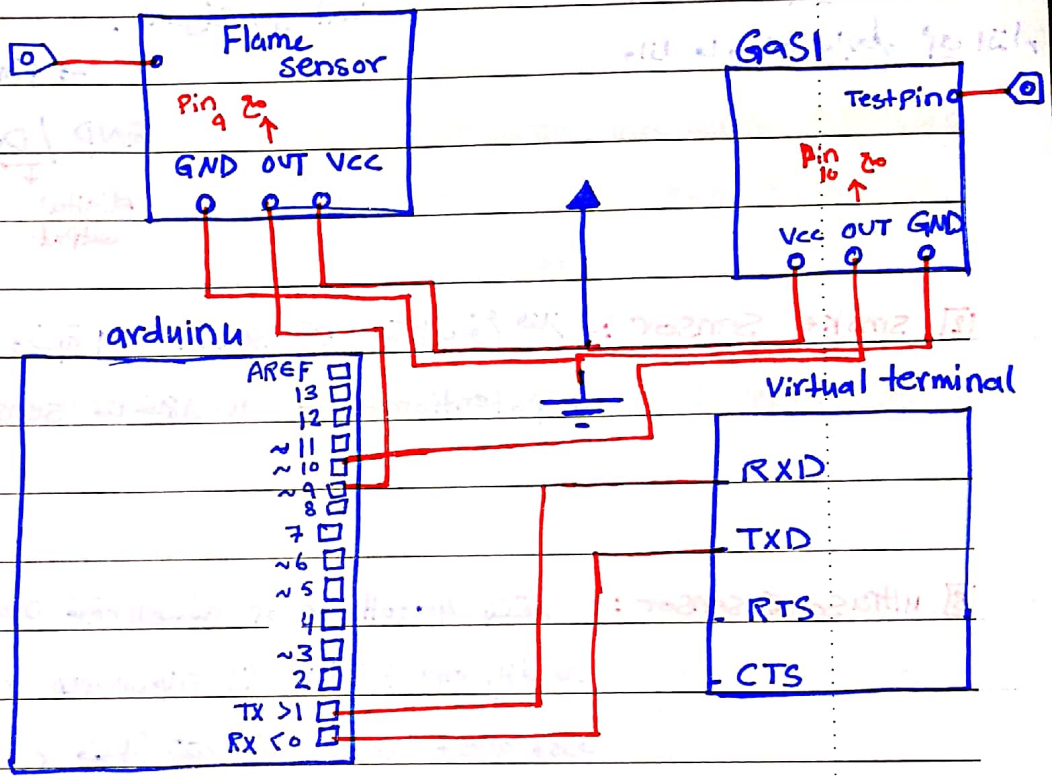
بس لو بدي أعرف مسافة
الذهاب والاياب بقسم على 2 .

Subject Date No.

هلا بنفتح البروتوس ، بنكبس new design ثم بنضيف الأشياء يلي رح نستعملها وبنضيف هاي الفلات

* SIMULINO UNO / flame sensor / gas sensor MQ-2 / logic state / Power / ground

هلا بنبلش نرسمهم عالشاشة ، وعشان نحوفن عن الـ serial monitor بنعمل virtual terminal



كل سنسر بنحفظ عليه double click وبنضيف عليه hex file .

```
int flame_snsr = 9 ;
int smoke_snsr = 10 ;

Void setup()
{ pinMode (flame_snsr , INPUT) ;
pinMode ( smoke_snsr , INPUT) ;
Serial. begin (600) ;
}
```

} بعرّف أرقام الـ Pins اللي شبكنا عليهم السنسورات.

} عرّفنا هدول الـ pins عالسي اللهم input لأنه من خلالهم رح تيجي الداتا

Subject Date No.

```
void loop()
{ int flame_value = digitalRead(flame_snsr);
  int smoke_value = digitalRead(smake-snsr);

  if (flame_value == HIGH)
  { serial.println("warning!");
    serial.println("flame detected!");
  }
  if (smake_value == High)
  { Serial.println("warning!");
    Serial.println("smake detected!");
  }
  if (smak_value == Low && flame_value == Low)
  { serial.println("SAFE");
  }
```

digital read بنقل

. Show verbose output ← references ← file نيكس على during

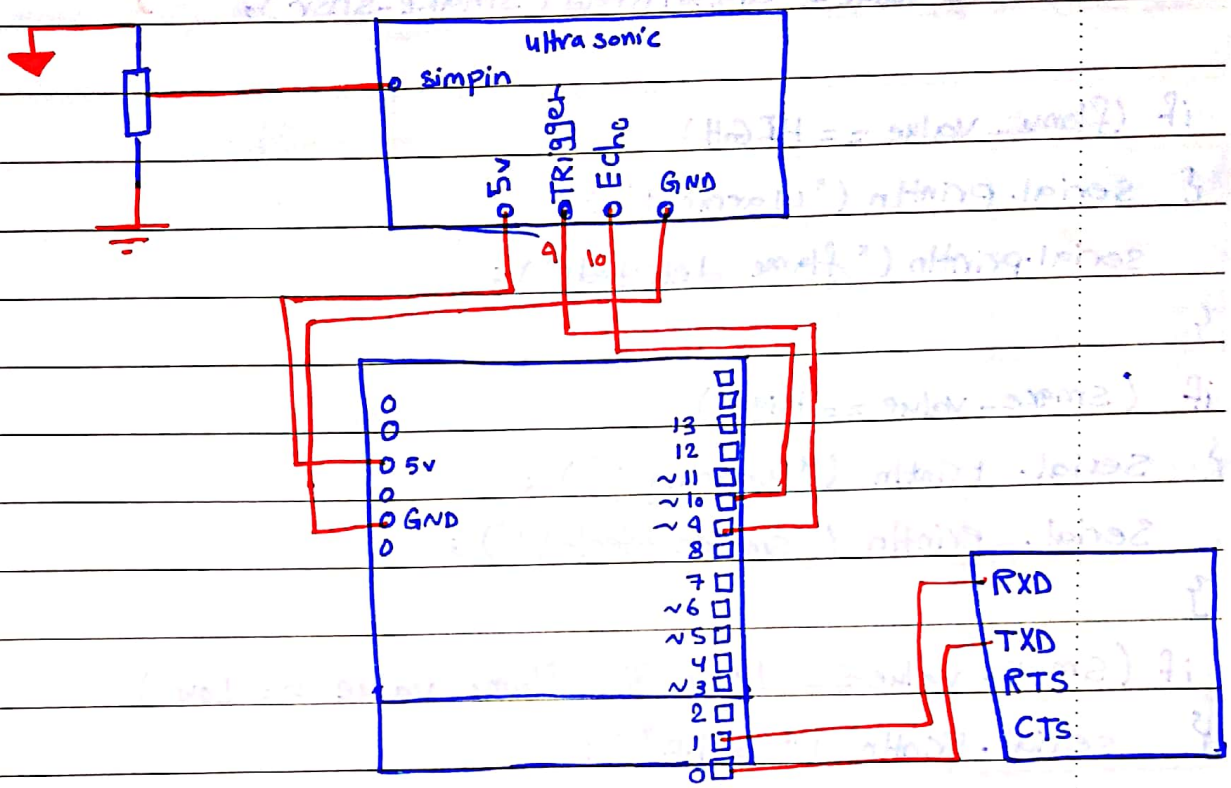
ثم بنقل run وبنوخد نسخ الـ Path وبنلزقه على الأردوينو لي بالبروتوس.

CamScanner

Subject Date No.

هنا بنا نشتغل Ultrasonic sen.. وهو بنتاج هاي الأشياء :-

Ultrasonic / pot-hg / SIMUINO UNo / logic State :




```
Const int trignPin = 9;
const int echo Pin= 10;
long duration;
int distance;

Void setup()
{ PinMode( trigPin, OUTPUT);
  PinMode (echoPin, INPUT);
  Serial.begin (6900);
}
```

NOTEBOOK

Subject Date No.

```
void Loop ()
{ digitalWrite (trigPin, LOW);
  delayMicroseconds(2);

digital write (trigPin, HIGH);
delay Micro second (10);
digital write ( trigPin, LOW);

duration = PulseIn ( echoPin, HIGH);
distance = duration * 0.034 / 2 ;

Serial. print( "Distance :");
Serial. println (distance);
}
```


CamScanner

Subject Date No.

write a code to design of an arduino based home fire/ -: كتابة كود لتصميم
gas alarm system. when detect a flame or gas; the buzzer
alarm become on, the LED(1) on when detect a flame, D2
led on when detect gas, alert (using serial communication)
with "warning" message, in the normal state, the leDs
are off and "safe" statement. message.

```
int flame_snsr = 11;
int gas_snsr = 10;
int D1 = 8;
int D2 = 9;
int buzzer = 7;

void setup()
{ PinMode (flame_snsr, INPUT);
  "       (gas_snsr, INPUT);
  "       (D1, OUTPUT);
  "       (D2, OUTPUT);
  "       (BUZZer, OUTPUT);
  Serial.begin (9600);
}
```

حسب واحدة

N O T E B O O K

Subject Date No.

```
Void loop()
{ int flame_value = digitalRead(flame_snsr);
  int gas_value = digitalRead(gas_snsr);

  if (flame_value == HIGH || Gas_value == HIGH)
  { digitalwrite(8,1);
    Serial.println("Warning");
    if (flame_value == HIGH)
    { digitalWrite(D1,1);
      digitalwrite(D2,0);
    delay(1000);
}
  if (gas_value == HIGH)
  { digitalwrite(D1,0);
    //          (D2,1);
}
  else
{ serial.println("SAFE");
  digitalwrite(D1,0);
    //          (De/0);
}}
```

NOTEBOOK

CamScanner

Subject exp(8) :- arduino and motors Date No.

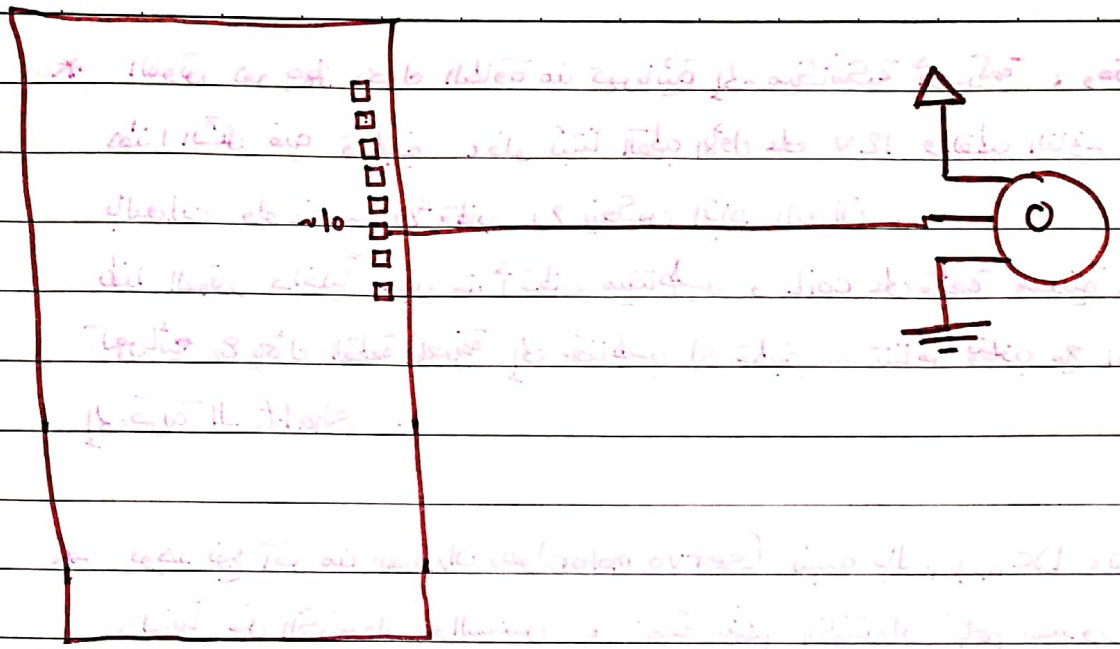
* الموتور هو جهاز يحول الطاقة من كهربائية إلى ميكانيكية أو حركية ، ومن أشكاله Dc motor هذا الشكل فيه قطبين ، فلو ثبتنا القطب الأول على 12V والقطب الثاني على الأرض ، ح يبدأ بالدوران ولو عكسنا الأقطاب ح ينعكس اتجاه الدوران .

هذا الموتور داخلياً مبني من أقطاب ميغناطيس و coil على قطعة معدنية ـ فعند وصول تيار كهربائية ح يتحول القطعة المعدنية إلى مغناطيس له قطبين ... تتنافر وتجذب مع المختلف لها ، مما يؤدي إلى حركة الـ shaft ..

* يوجد نوع آخر من الموتورات وهو (servo motor) شبيه بالـ Dc motor ، لكن فيه عناصر اضافية داخلياً مثل الكونترولر والسينسور ، حيث يتحكم الكونترولر بكم سيدور السيرفو موتور ، لأنه السيرفو موتور بزوايا محددة مش دوران كامل .. بعض الأنواع بتدور من 0 - 180° أو 0 - -180° أو 0 - 360° والعكس ، والذي يحدد انه الموتور وصل الزاوية المطلوبة أولسا ، هو السينسور .

* كيف بنتحكم بهاي الزوايا ؟ السيرفو موتور بوخد من الـ Pulse ، حسب الـ width تبع الـ Pulse اللي بنطيه إياها ح يتحرك بزوايا محددة ... اذا كانت الـ width تبعتها تساوي 1ms ح يتحرك الموتور بأقل زاوية ممكنة 180 ، 360 ... أما 2ms بتوصله لزاوية 180 وبنتصرف بنفس الطريقة أصغر Puls وأكبر Puls ح توصله لزاوية صفر ...

Subject Date No.



نعرّف الـ Pin بالـ Setup على أساس إنه output ثم نكتب Pin

```
digitalWrite(10, HIGH);
delay (3) ;
digitalWrite (10, LOW);
delay (20);
```

هتخلي السيرفو ع يضل تقريباً عند الزاوية صفر .

لو عدلنا الـ delay إلى 5 بدل 3 ع يصير 180° .

لو عدلنا الـ delay إلى 1 ع يصير -180° .

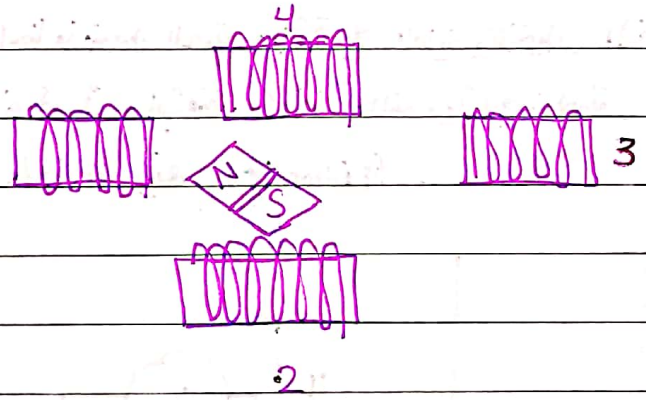
CamScanner

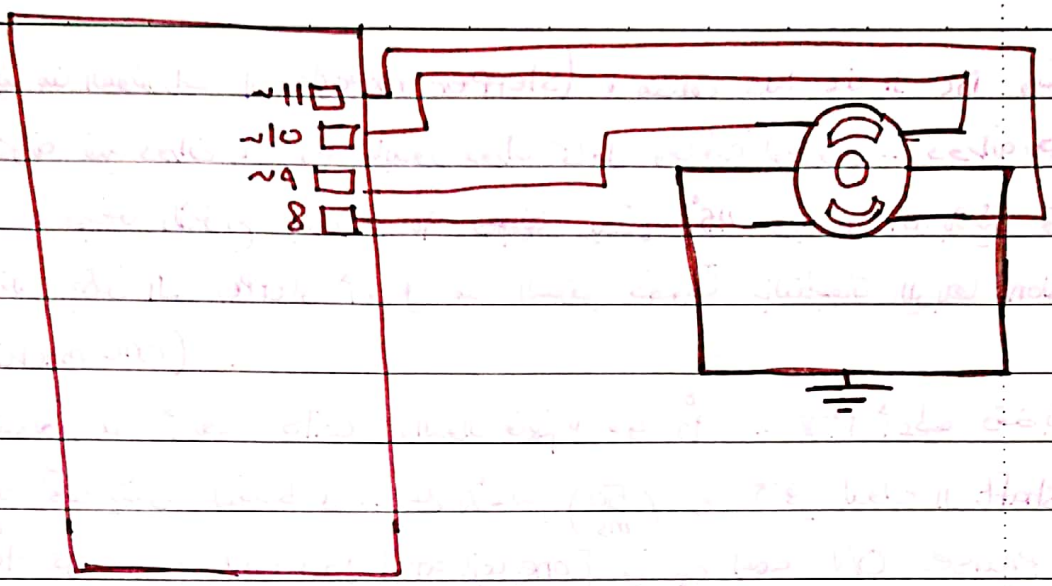
Subject Date No.

* عنا نوع جديد من الموتورات اسمه (Stepper motor) ، مختلف كليا عن الـ DC وشبيه تقريباً بالسيرفو كونه فيه دوران ، لكنه مابدور دوران كامل وما فيه كونترولر... دورانه بعتمد على الـ steps ، وبعض الأنواع بتدور بـ 90° وبعضها 45° وبعضها أقل بكثير قد تصل إلى 1° هاي الشغلة بتخلّي الـ Stepper أفضل من السيرفو خصوصاً بالتطبيقات اللي بدها Precision عالي (زي الـ cnc machine) .

طبعاً هذا الموتور من أربعة coils والدوران فيهم كل مرة 90° ... وزمان أعطيه وقت بكافئ الأربعة coils عشان يلحّق يتكون المغناطيس .. على الأغلب (50 ms) وقت كافي لدوران الـ shaft بشكل جيد بحيث بمشي 90° كل مرة عن طريق تفعيل [one coil at a time] اسمه one phase ON full stepping .

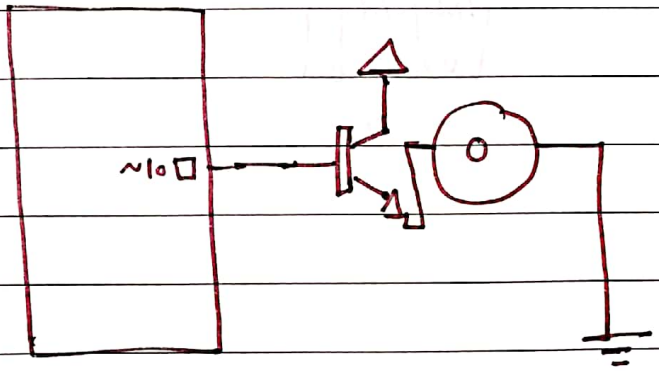
في عنا حالة ثانية ، حيث نفعّل 2 phase مع بعض ، كل phase فيهم رح يجذب المغناطيس اللي على الـ shaft اطرفه ، الـ phase الثاني بلغي تفعيل الـ coil رقم ٢ وبفعل الـ coil رقم 4 عشان ينقل حركة الـ shaft من منتصف المسافة بين ١و٢ إلى منتصف المسافة بين ١و4 وهكذا





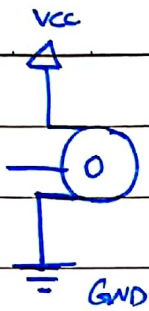
الحالة الأخيرة انه ندمج الحالتين السابقتين مع بعض بحيث نخلي (half stepping) يعني اذا كانت الموتر بيشتغل على 90 step بقدر نخليه يصير يشتغل على 45 اللي تعني الـ 90 .

آخر اشي ، احنا ما بنوصل الموتور مباشرة مع الماكروكونترولر لأنه احتمال يسحب كمية كبيرة من التيار تؤدي إلى حرقه أو انه أصلاً ما بشتغل على نفس الفولتية تبعت الموتور ، فلذلك بنستخدم هاي التوصيلة المناسبة باستخدام switch بنوع tip transistor .



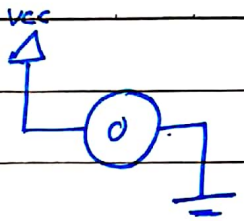
Subject Date No.

Servo motor :-



0° → 1ms
180 → 5ms
90° → 3ms

Dc motor :-



stepper motor :-

* one-phase (on full-stepping) 90°
45 / 135 / 225 / 315
1 / 2 / 3 / 4 / 1 ---

* two-phase (Full stepping) 90°
90 / 180 / 270 / 360
1+2 / 2+3 / 3+4 / 4+1 / ---

* one-two phase (on half stepping) 45°
45 / 90 / 135 / 180 / 225 / 270 / 315 / 360 ...
1 / 1+2 / 2 / 2+3 / 3 / 3+4 / 4 / 4+1 / 1 ---

- pinMode(pin, mode); (mode: input / output)
- digitalRead(pin);
- digitalWrite(pin, value); (value: high / low)
- delay(# millisecond) / delayMicroseconds(#);
- serial.begin(9600);
- serial.Print("---");
- serial.println("---");

NOTEBOOK