تصميم الدارات الالكترونية
باستخدام المتحكمات الصغرية

إعداد : م. حسام الوقائي

- ✓ المستوى الأول

مقدمة

يعد التطور الكبير الذي يشهده العالم في وقتنا الحاضر في شتى المجالات و خصوصاُ في المجال المعلوماتي و التقني من الأمور التي غيرت و بشكل كبير الحضارة الإنسانية التي مرت عبر التاريخ. ولعبت ثورة الدارات الالكترونية المتكاملة الجزء الأكبر في هذا التطور لما استطاعت أن تقدمه من إنجازات ما كان للإنسان أن يحلم بها سابقاً. فلو نظرنا من حولنا : البيوت – المحلات التجارية – المصانع – المزارع – وسائل النقل ..... نجد أنها تكاد لا تخلو من التكنولوجيا الالكترونية و التي تتمثل بالحساسات – أجهزة التشغيل – الإظهار – القيادة – الإنذار.....

تعد عملية تصميم الدارات الالكترونية التقليدية من الأمور التي يجد صعوبة في إنجازها الكثير من الناس و حتى المختصين في هذا المجال ، لهذا كان الحل من خلال استخدام دارة متكاملة تعرف بالمتحكم الصغري التي تقدم الحل الأبسط و الأمثل للدارة المطلوب تصميمها. المتحكم الصغري يعتبر بمثابة حاسب صغير يحتوي على معالج و ذواكر و عدادات وقد يحتوي على مبدلات ADC و وحدات اتصال تسلسلية..... و كل ذلك مصنع ضمن شريحة واحدة. وهذا ما يقدم للمصمم انتقالاً من الإجراء المبني على عناصر الكترونية Hardware إلى إجراء بسيط مبني على برمجة software ، وهذا ما يضفي المرونة و السهولة و الشمولية و الفاعلية.

أصبحت المتحكمات الصغرية جزءاً رئيسياً لأي جهاز الكتروني و إليك بعضاً من تطبيقاتها و التي نلتمسها في حياتنا اليومية : اللوحات الإعلانية – المصاعد – الأجهزة الطبية – أجهزة الإنذار – أجهزة التحكم – أجهزة القياس– التطبيقات الصناعية – التطبيقات العسكرية..... وغير ذلك من التطبيقات التي لا حصر لها.

إن ما يجعل المتحكمات الصغرية متداولة حالياً في كثير من المشاريع العملية و خصوصاً في مشاريع التخرج لدى طلاب الهندسات هي أنه يمكن من خلالها تصميم دارة أي مشروع و بأقل تكاليف و أفضل نتائج و أداء.

أتمنى أن يقدم كتاب (تصميم الدارات الالكترونية باستخدام المتحكمات الصغرية) للمبتدئين الخطوة الأولى لكل من أحب تعلم عالم المتحكمات الصغرية المشوق و المثير. و لا تبخلوا علي بالأسئلة و الاستفسارات و النقد البناء.

والله ولي التوفيق

م. حسام الوفائي

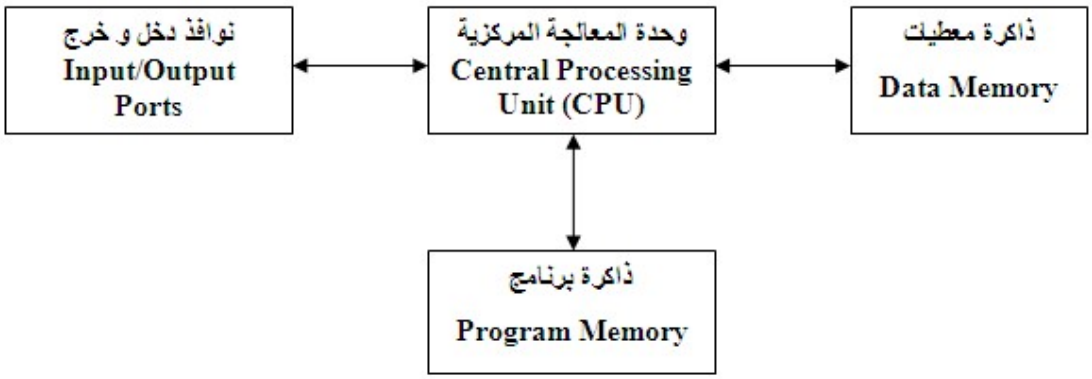
حمص 1/7/2013

Hussam.wafai@hotmail.com

الفهرس

النظام الحاسوبي

١- مكونات النظام الحاسوبي الأساسية



١-١- وحدة المعالجة المركزية Central Processing Unit (CPU)

تقوم وحدة المعالجة المركزية CPU بمعالجة و تنفيذ التعليمات و الأوامر المخزنة في ذاكرة البرنامج Program memory. قد تكون هذه التعليمات عبارة عن عمليات حسابية (جمع ، طرح ، ضرب ....) أو عمليات منطقية. قد تطلب هذه التعليمات من وحدة المعالجة قراءة معطيات من ذاكرة تخزين Data memory أو الكتابة فيها. و قد ترشد هذه التعليمات وحدة المعالجة بقراءة معطيات من نوافذ الدخل و إخراج معطيات على نوافذ الخرج.

١-٢- الذواكر Memories

الذواكر تعتبر الجزء الذي يتم به تخزين البيانات. تستخدم لأهداف متعددة في النظام الحاسوبي فقد يتم بها تخزين برنامج عمل النظام و تعرف هذه الذاكرة بذاكرة البرنامج Program memory ، من الممكن أن تستخدم الذاكرة لتخزين معلومات متنوعة كنواتج عمليات حسابية أو معطيات نوافذ الدخل و تعرف هذه الذواكر بذواكر المعطيات Data memory.

الأنواع الرئيسية للذواكر الموجودة في النظام الحاسوبي :

أ – ذواكر دائمة :وهي ذواكر لا تفقد معطياتها أثناء انقطاع التغذية عنها و لها عدة أشكال :

١. ذاكرة القراءة فقط Read Only Memory (ROM) : يتم الكتابة على هذه الذاكرة مرة واحدة فقط . أي أنه بعد تخزين البرنامج لا يمكننا مسحه أو تخزين برنامج آخر ، وتصبح صالحة للقراءة فقط.

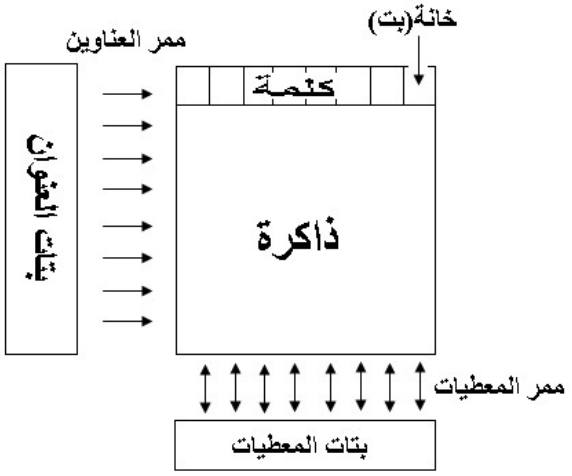
٢. ذاكرة قابلة للمحي عن طريق الأشعة فوق البنفسجية EPROM.

٣. ذاكرة قابلة للمحي والكتابة كهربائياً EEPROM. تعتبر بطيئة و يعتمد عدد مرات الكتابة عليها و المحو منها على طبيعة التصنيع.

٤. . ذاكرة قابلة للمحي والكتابة كهربائياً FLASH. تعتبر أسرع و أسهل استخداماً من EEPROM.

ب- **ذواكر متطايرة :** وهي ذواكر تفقد معطياتها أثناء انقطاع التغذية عنها و تتمثل بذاكرة الوصول العشوائي RAM.

أصغر وحدة تخزين في الذاكرة تعرف بالخانة أو البت تستطيع تخزين قيمة واحدة (0 أو 1) ، تجتمع مجموعة من الخانات (البتات) لتشكل ما يعرف بالموقع location أو الكلمة word. لكل موقع عنوان وحيد يختلف فيه عن أي موقع آخر. يتم التعامل مع الذاكرة سواء للقراءة أو الكتابة من خلال عنونة الموقع المطلوب وهذا ما يتم من خلال ما يعرف بخطوط العنونة ، أما القيمة المقروءة أو المراد تخزينها فتظهر على ما يعرف بخطوط المعطيات.



بفرض أننا أردنا كتابة كلمة في الذاكرة لابد أن نعلم:

١- عنوان تلك الكلمة – أي في أي موضع يجب أن نضعها – و نرسل ذلك العنوان عن طريق خطوط أو ممر العناوين.

٢- قيمة تلك الكلمة – أي ما تحتويه هذه الكلمة من واحدات و أصفار – و نرسل تلك المعطيات عن طريق خطوط أو ممر المعطيات.

بفرض أنه لدينا خط عنونة وحيد s_0 . هذا الخط يمكن أن نرسل عليه 0 أو 1 وبالتالي لدينا موقعين فقط في تلك الذاكرة أي أن سعتها 2 كلمة.

- بفرض أنه لدينا خطا عنونة s_0 و s_1 . هذان الخطان يمكن أن نرسل عليهما

s_0	s_1
0	0
0	1
1	0
1	1

وبالتالي لدينا أربع مواقع في تلك الذاكرة أي أن سعتها 4 كلمة.

و بشكل عام فإن سعة الذاكرة

$$2^n \text{ word}$$

n : هي عدد خطوط العنونة.

يشار إلى الكلمة بالبايت إذا كانت الكلمة مؤلفة من 8 خانات.

عدد خطوط المعطيات يساوي عدد خانات الكلمة.

مثال ذاكرة تملك 11 خط عنونة و 8 خطوط معطيات . حدد سعتها بالبايت و البت.

سعة الذاكرة بالبايت = $2^n = 2^{11} = 2048$ byte

سعة الذاكرة بالبت= $2048 \times 8 = 16384$ bit

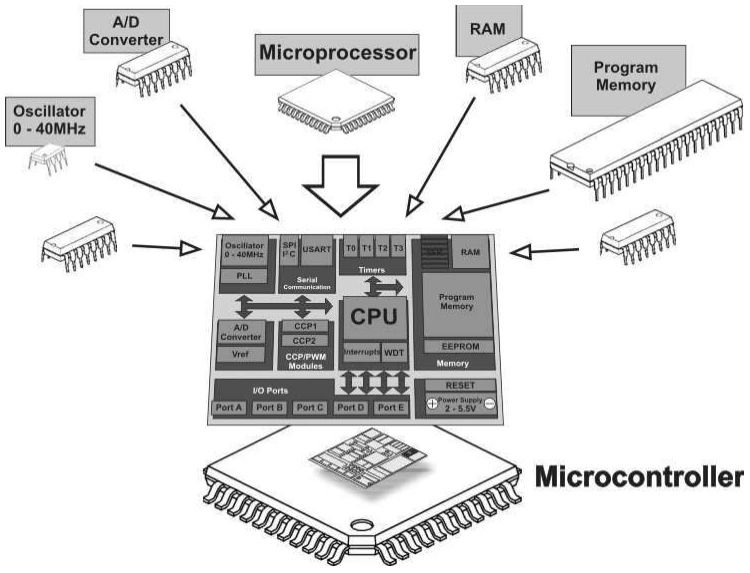
تحتوي الذواكر إضافة لتلك الخطوط على خط تفعيل للذاكرة CE و التغذية V_{cc} و الأرضي. إضافة لخط الكتابة إلى الذاكرة و خط القراءة من الذاكرة.

١-٣- ما هو المتحكم الصغري Microcontroller ؟

المتحكم الصغري هو عبارة عن حاسب صغير مصنع على شريحة واحدة يحتوي على :

- معالج صغري microprocessor.
- ذاكرة برنامج و هي ذاكرة قراءة قابلة للبرمجة (ROM,EPROM,EEPROM,FLASH).
- ذاكرة وصول عشوائي RAM.
- منافذ دخل و خرج I/O ports.
- وحدات أخرى مثل المؤقتات Timers ، مبدلات تمثيلية رقمية ADC.

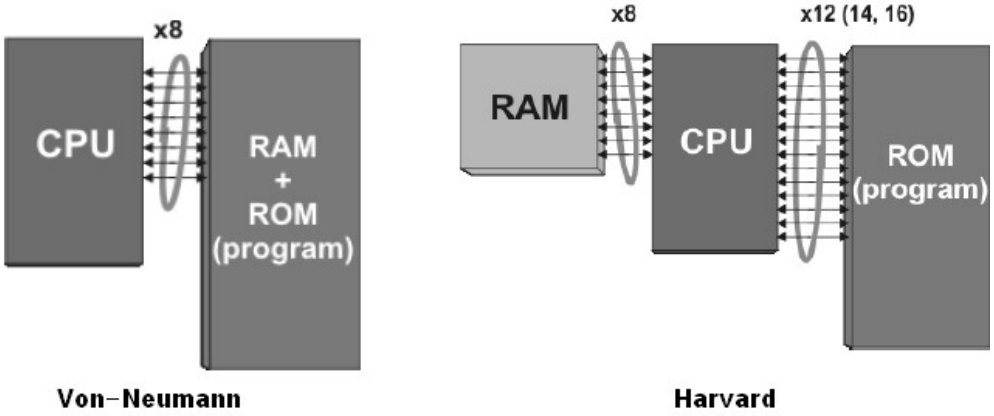
يقوم المبرمج بوضع برنامجه في ذاكرة البرنامج. ليقوم المعالج بتنفيذ تعليمات البرنامج كإدخال قيم معينة ، إخراج قيم على نوافذ الخرج و تخزين النتائج في ذواكر RAM أو EEPROM و هذا ما سنتعلمه لاحقاً



تختلف طريقة ربط وحدة المعالجة المركزية مع الذواكر في المتحكمات الصغرية بإحدى الطريقتين التاليتين:

☒ **بنيـة Von-Neumann** : المتحكمات التي تستخدم هذه البنية لها ذاكرة واحدة يتم بها تخزين المعطيات و البرنامج ، ترتبط مع المعالج من خلال مسار معطيات بثمانية خطوط . لهذا فإن المعالج إما قد يقرأ تعليمات (أوامر) أو قد يقرأ و يخزن معطيات ولا يستطيع أن ينفذ كلا العمليتين معاً. بالتالي فإن مثل هذه البنية تتسم بالبطء و عدم الفاعلية.

☒ **بنية Harvard** : المتحكمات التي تستخدم هذه البنية لها ذاكرتان منفصلتان : ذاكرة برنامج و ذاكرة معطيات. يتصل المعالج مع ذاكرة المعطيات مع مسار بثمانية خطوط ، و يتصل مع ذاكرة البرنامج من خلال مسار آخر عبارة عن عدة خطوط (12, 14 or 16) . لهذا فإن المعالج يستطيع أن يقرأ التعليمات من ذاكرة البرنامج و أن يتخاطب مع ذاكرة المعطيات بنفس الوقت وهذا ما يزيد من السرعة و الأداء.



يتم برمجة المتحكمات الصغرية من خلال لغة الاسمبلي Assembly وهي عبارة مجموعة من التعليمات و الأوامر التي يفهمها معالج المتحكم الصغري ولا يستطيع فهم غيرها – لغة المعالج – .

يوجد بنيتين مختلفتين يتم من خلالها تصميم المتحكم الصغري اعتماداً على التعليمات التي يفهمها :

☒ **RISC (Reduced Instruction Set Computer)** : : المتحكمات التي تعتمد على هذه البنية تستخدم تعليمات أسمبلي لا تزيد عن 35 تعليمة وهي تعليمات بسيطة (جمع ، طرح ، نسخ ...) ، العمليات الأكثر تعقيد يتم بناؤها انطلاقاً من هذه التعلميات البسيطة (فمثلاً الضرب يتم إنجازه انطلاقاً من عدة تعليمات جمع) . عدد التعليمات المنخفض يؤدي إلى تقليل العتاد و منه إلى انخفاض عدد الترانزستورات المشكلة للمعالج و بالتالي تخفيف الضياعات الكهربائية و انخفاض درجة حرارة المعالج وسعره.

☒ **CISC (Complex Instruction Set Computer)** : المتحكمات التي تعتمد على هذه البنية تستخدم ما يصل إلى 200 تعليمة أسمبلي. زيادة عدد التعليمات يؤدي إلى زيادة تعقيد العتاد و منه إلى زيادة عدد الترانزستورات المشكلة للمعالج و بالتالي تزداد الضياعات الكهربائية و تزداد درجة حرارة المعالج وسعره.

١-٤- متحكمات شركة Microchip

تنتج شركة Microchip متحكمات تعرف باسم PIC (Peripheral Interface Controllers). يتميز متحكم PIC عن غيره من المتحكمات بالعمل في ظروف تشويش عالية مثل أماكن وجود المحركات حيث يتوافر معظم إنتاج الشركة للصناعات العسكرية و التجارية.

تستخدم متحكمات PIC بنية Harvard RISC (Reduced Instruction Set Computer) ، و تتوافر بمواصفات ومزايا عديدة و تختلف فيما بينها بما يلي :

❖ نوعية ذاكرة البرنامج

C : EPROM
CR : ROM
CE : EEPROM
F : FLASH
HV : High Voltage (15V)
LF : Low Voltage Flash
LC : Low Voltage EEPROM
LCR : Low Voltage ROM

❖ سعة ذاكرة البرنامج

0.5–1K
2–4K
8–16K
24–32K
48–64K
96–128K

❖ عدد أرجل الدخل و الخرج

❖ ملحقات خاصة

ADC
CAN
USB
USART
I2C
SPI
Ethernet
Motor Control
Radio Frequency

تم تصنيف متحكمات PIC ضمن العائلات التالية:

١. عائلة PIC12CXXX/PIC12FXXX

لهذه العائلة متحكمات بعدد أطراف 8 pin و بعرض ذاكرة برنامج 12bit أو 14 bit.و أهم هذه المتحكمات:

PIC12F675 , PIC12F629 , PIC12C508A , PIC12C509A , PIC12C671 , PIC12C672 , PIC12C518 , PIC12C519 , PIC12C673 , PIC12C674

٢. عائلة PIC16C5X

لهذه العائلة متحكمات بعدد أطراف 8 pin أو 28 pin و بعرض ذاكرة برنامج 12 bit. و أهم هذه المتحكمات PIC16C54 , PIC16C55 , PIC16C56 , PIC16C57

٣. عائلة PIC16CXXX/PIC16FXXX

لهذه العائلة متحكمات بعدد أطراف 14 pin أو 18 pin أو 28 pin أو 40 pin و بعرض ذاكرة برنامج 14 bit. و أهم هذه المتحكمات

PIC16F627 , PIC16F628 , PIC16F630 , PIC16F648 , PIC16F676 , PIC16F83 , PIC16F84 , PIC16F87 , PIC16F870 , PIC16F871 , PIC16F873 , PIC16F874 , PIC16F876 , PIC16F877 , PIC16F88, PIC16C61 , PIC16C62 , PIC16C63 , PIC16C64 , PIC16C65 , PIC16C66 , PIC16C67 , PIC16C72 , PIC16C73 , PIC16C74 , PIC16C76 , PIC16C77

٤. عائلة PIC17CXXX

هذه العائلة لها عرض ذاكرة برنامج 16 bit.

٥. عائلة PIC18CXXX/PIC18FXXX

لهذه العائلة متحكمات بعدد أطراف 18 pin أو 28 pin أو 40 pin و بعرض ذاكرة برنامج 16 bit. لهذه العائلة مزايا عديدة : سرعة العمل العالية – سعة ذاكرة البرنامج كبيرة – الوحدات الداخلية كثيرة و التي تساعد في تقليل الوحدات الخارجية الملحقة مما يوفر التكلفة والطاقة المستهلكة. يوجد الكثير من المتحكمات التابعة لهذه العائلة.

٦. عائلة PIC24FXXX

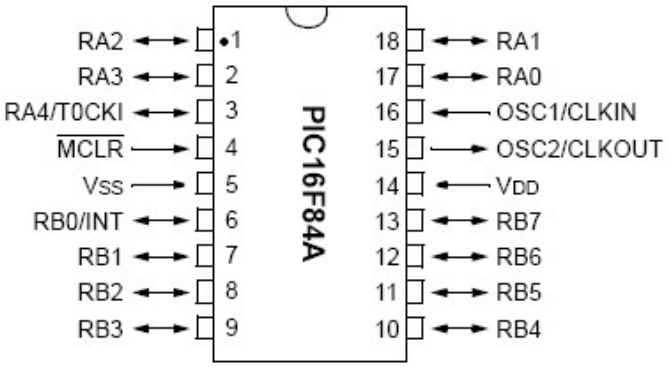
تتميز هذه العائلة بأنها SMD و بعدد أطراف 64 pin أو 80 pin أو 100 pin و ذات سعة برنامج كبيرة جداً. و بعرض ذاكرة برنامج 16 bit.

٧. dsPIC30XXX/dsPIC33XXX

أهم ما يميز هذه العائلة أنها تدعم معالجة الإشارة الرقمية digital signal processing . و أهم هذه المتحكمات dsPIC30F4013

المتحكم الصغري PIC16F84A

٣-١- أقطاب المتحكم 16f84A:



❖ الأقطاب 17 , 18 , 1 , 2 , 3 تمثل النافذة A . وتعتبر نافذة دخل و خرج رقمية ، أي أنه من خلال هذه النافذة يمكن إدخال واحدات و أصفار منطقية (5v , 0v) أو إخراج واحدات و أصفار منطقية (5v , 0v). القطب رقم 3(RA4/T0CKI) بالإضافة لعمله كقطب دخل و خرج يمكن أن يكون مدخل نبضات للعداد TIMER0.

❖ الأقطاب 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 تمثل النافذة B . وتعتبر نافذة دخل و خرج رقمية. القطب رقم 6(RB0/INT) بالإضافة لعمله كقطب دخل و خرج يمكن أن يكون مدخل مقاطعة خارجية. الأقطاب , 13 10 , 11 12بالإضافة لعملها كأقطاب دخل و خرج يمكن أن تكون مدخل لمقاطعة خارجية في حالة تغير حالتها المنطقية.

❖ Vss أرضي

❖ VDD تغذية (5V).

❖ MCLR مدخل تصفير أساسي للمتحكم (إعادة إقلاع المتحكم) . بحالة العمل العادية لا بد أن تكون هذه الرجل موصولة إلى 5V و عند وصلها إلى 0V يتم إعادة إقلاع المتحكم .

❖ القطبان 15 , 16 يتم وصلهما إلى كريستالة خارجية ، هذه الكريستالة هي التي تحدد تردد عمل المتحكم ، بمعنى آخر تحدد سرعة المتحكم.

٣-٢- مزايا المتحكم 16F84A

- وحدة معالجة CPU ذات تقنية RISC و ترتبط مع الذواكر وفق تقنية Harvard.
- سعة ذاكرة البرنامج 1024 كلمة ، طول كلمة التعليمة 14 بت.
- سعة الذاكرة RAM : 68 بايت ، طول بايت المعطيات 8 بت.
- سعة الذاكرة EEPROM : 64 بايت ، وهي ذاكرة معطيات إضافية.

- ذاكرة مكدس بعمق 8 مستويات.
- سرعة التشغيل : يمكن وصل كريستالة حتى تردد 20MHz.
- 35 تعليمة فقط بلغة الاسمبلي.
- هناك أربعة مصادر للمقاطعة :

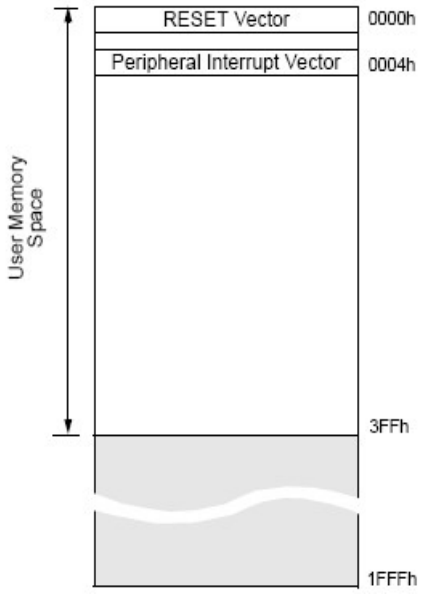
١. مقاطعة خارجية على القطب(RB0/INT) .
٢. مقاطعة تغير حالة الأقطاب (RB7,RB6,RB5,RB4).
٣. مقاطعة طفحان المؤقت TIMER0.
٤. مقاطعة انتهاء الكتابة في ذاكرة المعطيات EEPROM.

- 13 قطب دخل و خرج رقمي.
- تيار المصرف sink الأعظمية 25mA لكل قطب.
- تيار الخرج (مصدر) source الأعظمي 25mA لكل قطب.
- إمكانية حماية برنامج التشغيل.
- إمكانية الدخول في نظام الراحة SLEEP لتوفير الطاقة.
- ذواكر FLASH/EEPROM ذات تقنية CMOS المحسنة و التي لها طاقة منخفضة و سرعة عالية ،
- مجال تشغيل عريض V(2–5.5).

٣-٣- ذاكرة البرنامج Program Memory

- ذاكرة البرنامج

لها سعة 1Kword ، word = 14 bit.



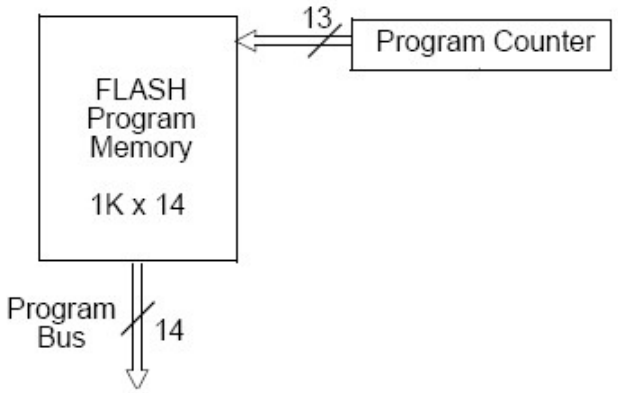
لاحظ أن المواقع من بعد 3FFh إلى 1FFFh غير موجودة فيزيائياً. و تبقى المواقع من 0000h إلى 3FFh (1024) هي المستخدمة.

في كل موقع يتم تخزين تعليمة أسمبلي واحدة ، أي أن كل تعليمة ترمز بـ ١٤ بت.

- عندما نقوم بعملية تصفير المتحكم (MCLR) يتم الانتقال إلى شعاع التصفير RESET Vector والذي عنوانه 0000h
- عندما تتحقق شروط المقاطعة يتم الانتقال إلى شعاع المقاطعة Peripheral Interrupt Vector والذي عنوانه 0004h و الذي يحوي عنوان برنامج المقاطعة.

- عداد البرنامج program counter

هو عداد ذو 13 بت يتصل إلى خطوط عنونة ذاكرة البرنامج ، و بالتالي يستطيع أن يعنون ذاكرة لها سعة $2^{13} = 8192 =$ 8Kword .



في متحكم 16F84A طول عداد البرنامج 13 بت بالرغم أن ذاكرة البرنامج سعتها 1Kword حيث تأخذ الخانات 10,11,12 قيمة 0 دائماً و الخانات العشرة المتبقية تستخدم لعنونة الذاكرة.

- زمن تنفيذ تعليمة الاسمبلي :

زمن تنفيذ تعليمة الاسمبلي الواحدة (أو ما يعرف بزمن دورة الآلة Tcy) يساوي :

$$Tcy = \frac{1}{\text{(4/ تردد الكريستالة)}}.$$

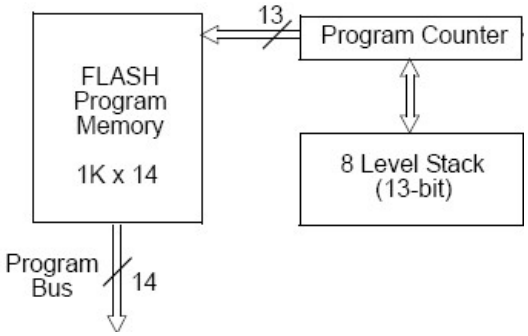
مثال ليكن لدينا كريستالة بتردد 4MHz احسب زمن تنفيذ التعليمة الواحدة المخزنة في ذاكرة البرنامج :

$$Tcy = \frac{1}{(4 \times 10^6 / 4)} = 1\ \mu s$$

٣-٤-المكدس Stack:

يخزن المكدس عنوان العودة إلى البرنامج الرئيسي عند استخدام برنامج فرعي أو الانتقال إلى برنامج مقاطعة.

عدد خانات الموقع الواحد = عدد ممر العناوين = 13 bit.



مثال يوضح عمل المكدس

10 call 500
11
12
.
.
.
500
. برنامج فرعي
.
.
600 Return

عند استدعاء البرنامج الفرعي تتم العمليات التالية :

- يخزن في المكدس عنوان العودة للبرنامج الرئيسي و هو 11= 1 + 10 = PC +1 = stk(0)
- يخزن في عداد البرنامج عنوان البرنامج الفرعي 500 = PC و عندئذ تتم عملية القفز للبرنامج الفرعي.
- يزداد عداد البرنامج بمقدار واحد ويستمر PC = PC +1

عند السطر 600 توجد تعليمة Return ، أي العودة للبرنامج الرئيسي و فيه تتم العمليات التالية :

- يشحن العداد بالقيمة المخزنة في المكدس و هي 11 و بالتالي العودة للبرنامج الرئيسي PC = stk(0)
- يزداد العداد بمقدار واحد و يستمر بالزيادة منفذاً الأجزاء المتبقية من البرنامج.

للمكدس ثمانية مواقع أو مستويات و بالتالي هناك إمكانية باستدعاء ثمانية برامج فرعية متداخلة.

٢-٥- ذاكرة المعطيات RAM :

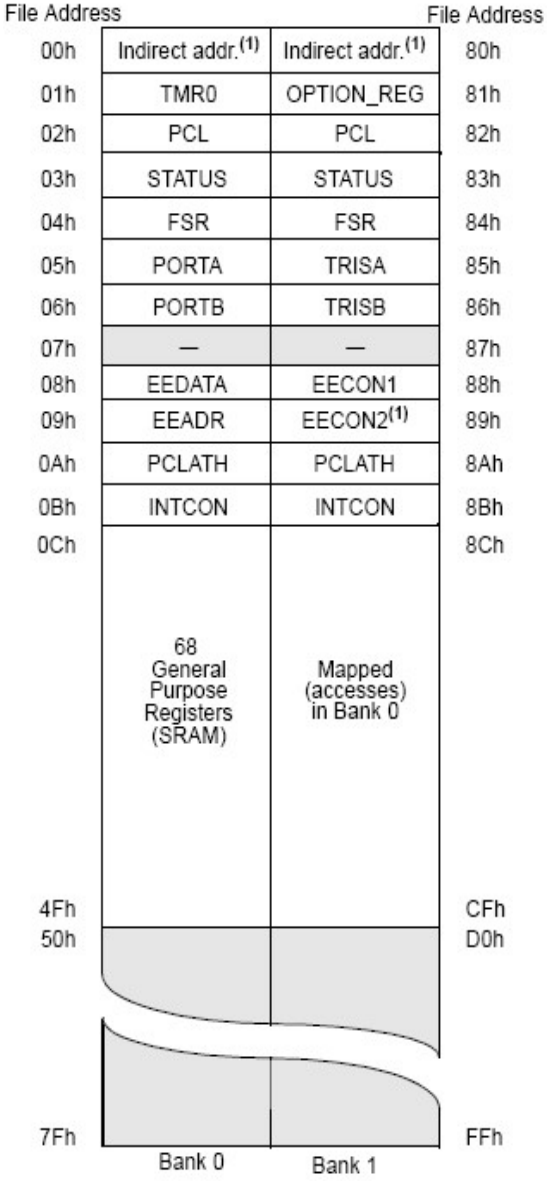
تقسم ذاكرة المعطيات في متحكم 16F84 إلى بنكين ، سعة كل منهما 128Byte .

البنك (و بشكل عام ذاكرة المعطيات) يقسم إلى ثلاثة حيزات :

١- **حيز مسجلات الوظائف الخاصة SFR:** عن طريق هذه المسجلات يتم التحكم بالشريحة كما سنرى لاحقاً.

٢- **حيز مسجلات الأغراض العامة GPR:** عن طريق هذه المسجلات يتم تخزين المعطيات (أرقام مثلاً).

٣- **حيز غير مستخدم.**



- مسجلات الأغراض الخاصة SFR

١- TRISA:

عنوانه 85h (البنك 1). هذا المسجل يحدد هل النافذة A نافذة دخل أم خرج.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
85h	TRISA	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	---1 1111	---1 1111

قيمة الخانة / الخانة	0	1
(TRISA0)0	الرجل أو القطب RA0 خرج	الرجل أو القطب RA0 دخل
(TRISA1)1	الرجل أو القطب RA1 خرج	الرجل أو القطب RA1 دخل
(TRISA2)2	الرجل أو القطب RA2 خرج	الرجل أو القطب RA2 دخل
(TRISA3)3	الرجل أو القطب RA3 خرج	الرجل أو القطب RA3 دخل
(TRISA4)4	الرجل أو القطب RA4 خرج	الرجل أو القطب RA4 دخل
(X)5	لا يهم	لا يهم
(X)6	لا يهم	لا يهم
(X)7	لا يهم	لا يهم

٢- TRISB:

عنوانه 86h (البنك 1). هذا المسجل يحدد هل النافذة B نافذة دخل أم خرج.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111

قيمة الخانة / الخانة	0	1
(TRISB0)0	الرجل أو القطب RB0 خرج	الرجل أو القطب RB0 دخل
(TRISB1)1	الرجل أو القطب RB1 خرج	الرجل أو القطب RB1 دخل
(TRISB2)2	الرجل أو القطب RB2 خرج	الرجل أو القطب RB2 دخل
(TRISB3)3	الرجل أو القطب RB3 خرج	الرجل أو القطب RB3 دخل
(TRISB4)4	الرجل أو القطب RB4 خرج	الرجل أو القطب RB4 دخل
(TRISB5)5	الرجل أو القطب RB5 خرج	الرجل أو القطب RB5 دخل
(TRISB6)6	الرجل أو القطب RB6 خرج	الرجل أو القطب RB6 دخل
(TRISB7)7	الرجل أو القطب RB7 خرج	الرجل أو القطب RB7 دخل

PORTA -٣

عنوانه 05(h) البنك 0.

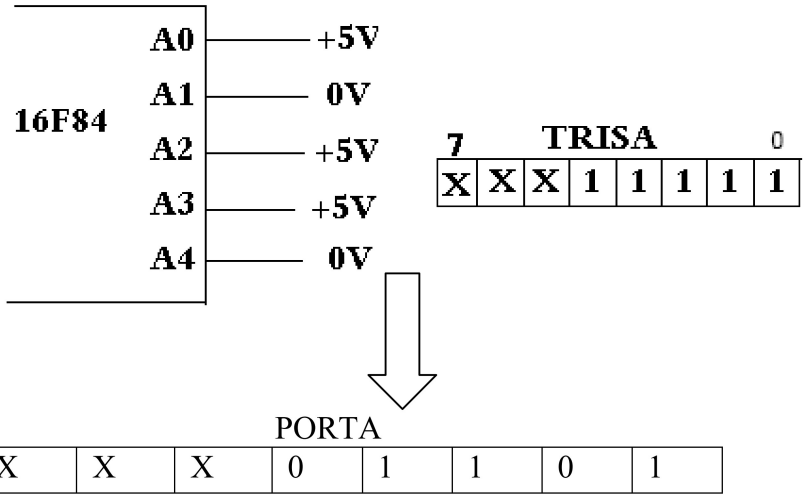
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other RESETS
05h	PORTA	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are unimplemented, read as '0'.

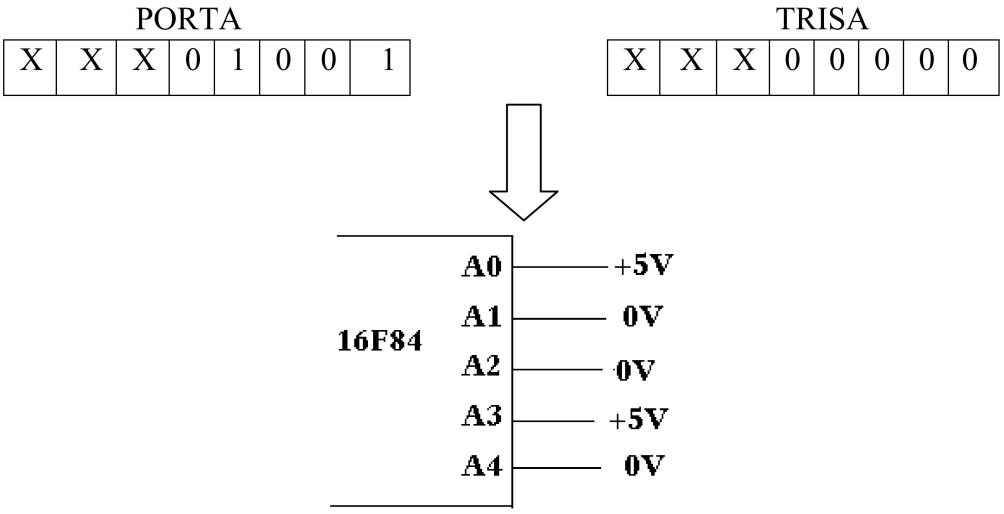
سنناقش وظيفة هذا المسجل وفق النقطتين التاليتين:

أ- بفرض أن قمنا بتهيئة النافذة A على أنها نافذة دخل عن طريق المسجل TRISA . عندئذ فإن المسجل PORTA سوف يحتوي على القيم المطبقة على النافذة A.

مثال



ب-بفرض أننا قمنا بوضع النافذة A على أنها نافذة خرج عن طريق المسجل TRISA . عندئذ فإن القيم التي نضعها في المسجل PORTA ستظهر عندئذ على أرجل النافذة A.



4- PORTB

عنوانه 06(h) البنك 0. هذا المسجل مشابه في عمله للمسجل PORTA.

5- TIMER0

المؤقت TMR0 هو عبارة عن مسجل بطول 8 bit ، عنوانه 01(h) البنك 0 ، يقوم بعد النبضات القادمة من الرجل A4 (عداد) أو النبضات القادمة من مهتز الكريستالة (مؤقت) و تتزايد قيمته باستمرار حتى تصل القيمة إلى الرقم 255 وهي القيمة الأعظمية له بعد ذلك يعاود البدء من جديد من الرقم 0 . يمكن إجمال خصائص TIMER0 بما يلي :

١. يقوم بعد النبضات القادمة من الرجل A4 (عداد) أو النبضات القادمة من مهتز الكريستالة (مؤقت) ويتم ضبط ذلك من خلال المسجل OPTION الذي سنشرحه لاحقاً.
٢. يمكن الكتابة فيه أو القراءة منه .
٣. اختيار الحافة الصاعدة أو الهابطة عند العد من خلال خانة موجودة في OPTION.
٤. يوجد معدل مقياس prescaler (مقسم ترددي) يتم ضبطه من خلال المسجل OPTION.
٥. يوجد مقاطعة طفحان المؤقت عندما ينقلب من FFh(255) إلى 00h .

٦- مسجل التحكم بالمقاطعات INTCON REGISTER

عنوانه 0Bh و 8Bh (البنك 0 و البنك 1)

- هناك أربعة مصادر للمقاطعة في 16f84 :

١. مقاطعة خارجية على القطب(RB0/INT) .
٢. مقاطعة تغير حالة الأقطاب (RB7,RB6,RB5,RB4).
٣. مقاطعة طفحان المؤقت TIMER0.
٤. مقاطعة انتهاء الكتابة في ذاكرة المعطيات EEPROM.

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

قيمة الخانة / الخانة	0	1
(GIE)7	يتم حجب كل المقاطعات	يتم تمكين كل المقاطعات
(EEIE)6	يتم حجب مقاطعة انتهاء الكتابة في الذاكرة EEPROM	يتم تفعيل مقاطعة انتهاء الكتابة في الذاكرة EEPROM

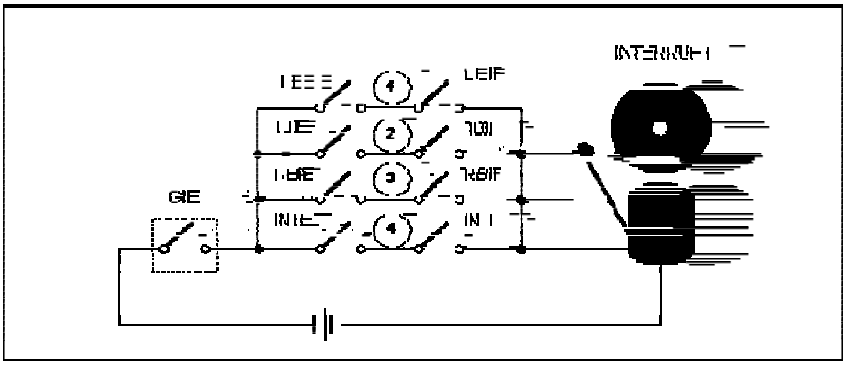
(TOIE)5	يتم حجب مقاطعة طفحان المؤقت	يتم تمكين مقاطعة طفحان المؤقت
(INTE)4	يتم حجب المقاطعة الخارجية على الرجل RB0	يتم تمكين المقاطعة الخارجية على الرجل RB0
(RBIE)3	يتم حجب مقاطعة تغير حالة الأقطاب(RB7,RB6,RB5,RB4)	يتم تمكين مقاطعة تغير حالة الأقطاب(RB7,RB6,RB5,RB4)
(TOIF)2	عند عدم حدوث طفحان المؤقت	عند حدوث طفحان المؤقت
(INTF)1	عند عدم ورود نبضة(حافة صاعدة أو هابطة) على الرجل RB0	عند ورود نبضة(حافة صاعدة أو هابطة) على الرجل RB0
(RBIF)0	عند عدم تغير حالة الأقطاب (RB7,RB6,RB5,RB4)	عند تغير حالة الأقطاب (RB7,RB6,RB5,RB4)

بالتالي لكي تتحقق المقاطعة لابد من توافر ثلاثة شروط :

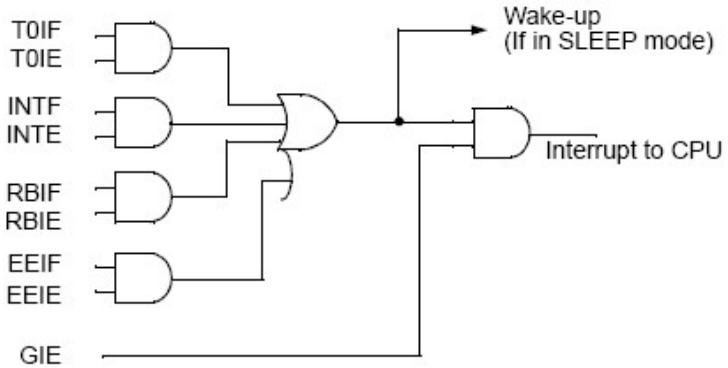
١- تفعيل بت المقاطعة العامة GIE=1.

٢- تفعيل البت الخاص بتلك المقاطعة.

٣- حدوث المقاطعة بحد ذاتها(تأثر المؤشر الخاص بها Flag).



Simplified outline of PIC 16F84 microcontroller interrupt



٧- مسجل الاختيار OPTION REGISTER

عنوانه 81h (البنك 1).

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

قيمة الخانة / الخانة	0	1
(RBPU)7	يتم تمكين مقاومات السحب على النافذةB	يتم إلغاء مقاومات السحب على النافذةB
(INTEDGE)6	يتم ضبط المقاطعة الخارجية على الرجل RB0 على الحافة الهابطة	يتم ضبط المقاطعة الخارجية على الرجل RB0 على الحافة الصاعدة
(TOCS)5	المسجل TMR0 سيعد نبضات داخلية ناتجة عن مذبذب الكريستالة (Fosc/4)	المسجل TMR0 سيعد نبضات خارجية مطبقة على الرجل RA4
(TOSE)4	العداد TMR0سيزداد عند الحافة الصاعدة	العداد TMR0 سيزداد عند الحافة الهابطة
(PSA)3	المقسم Prescaler سيكون تابع لـ TMR0	المقسم Prescaler سيكون تابع لـ WDT

يقوم المقسم Prescaler بتقسيم النبضات الداخلة إليه و يتم ضبط قيمة التقسيم من خلال البتات 2 , 1 , 0 (PS0,PS1,PS2) كما يلي:

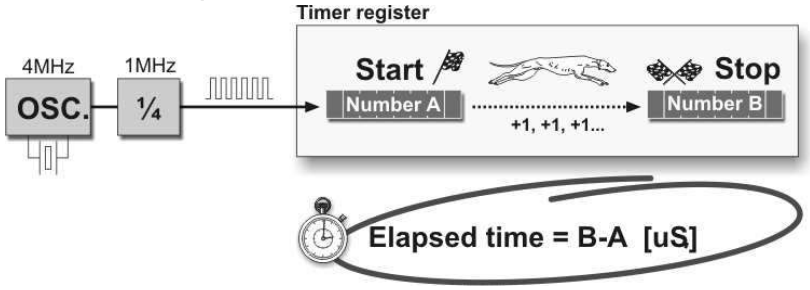
Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

مثال بفرض أن PS2 = 1 , PS1 = 0 , PS0 = 0

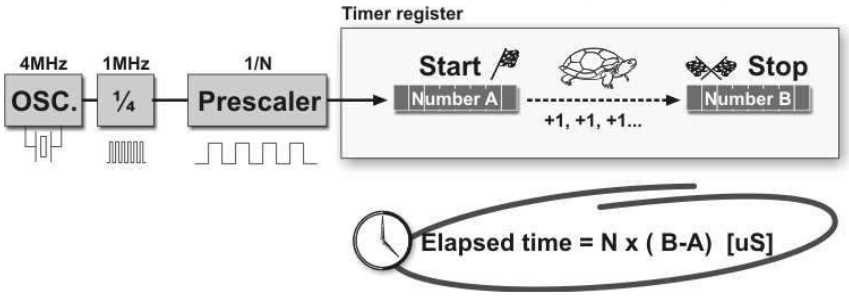
عندما يكون PSA=0 ، فإن كل 32 نبضة تأتي على الرجل RA4 أو من مذبذب الكريستالة (Fosc/4) سيزداد العداد TMR0 بمقدار واحد.

عندما يكون PSA=1 ، فإن كل 16 نبضة ناتجة عن مذبذب WDT سيزداد WDT بمقدار واحد.

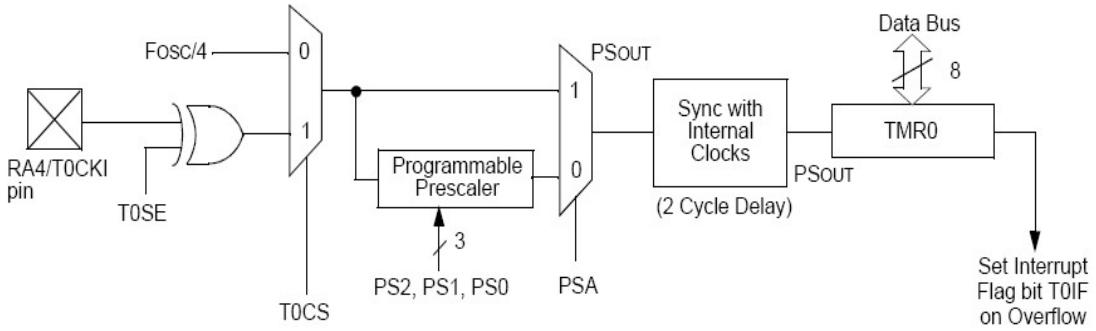
في حال عدم تشغيل Prescaler للمؤقت TIMER0 فإن النبضات التي ترددها $\frac{Fosc}{4}$ (Fosc هو تردد الكريستالة) ستطبق على المؤقت مما سيؤدي إلى أن يزداد المؤقت بشكل سريع :



في حال تشغيل Prescaler للمؤقت TIMER0 فإن النبضات التي ترددها $\frac{Fosc}{4*N}$ (Fosc هو تردد الكريستالة ، N قيمة المقسم Prescaler) ستطبق على المؤقت مما سيؤدي إلى أن يزداد المؤقت بشكل بطيء:



- كيفية عمل TIMER0



يعمل Timer0 كمؤقت أو عداد و يتم تحديد ذلك من خلال البت TOCS الموجود ضمن المسجل Option_Reg. فعندما يأخذ هذا البت صفر منطقي يعمل Timer0 كمؤقت و عندئذ يزداد Timer0 مع كل دورة تعليمة واحدة Fosc/4 (بدون المقسم المبدئي prescaler) بمعنى أنه سيعد النبضات القادمة من الكريستالة . في حالة الكتابة إلى هذا المسجل يتم إيقاف زيادة Timer0 لفترة مقدارها زمن دورتي تعليمة. وعندما يأخذ البت TOCS واحد منطقي يعمل Timer0 كعداد و عندئذ يزداد Timer0 مع كل حافة صاعدة rising edge أو حافة هابطة falling edge على الطرف RA4/TOCK1 . ويتم تحديد كيفية الزيادة (حافة صاعدة أو حافة هابطة) من خلال Timer0 Source Edge Select bit (TOSE) . فعندما يأخذ هذا البت صفر

منطقي يزداد Timer0 مع كل حافة صاعدة. وعندما يأخذ هذا البت واحد منطقي يزداد Timer0 مع كل حافة هابطة.

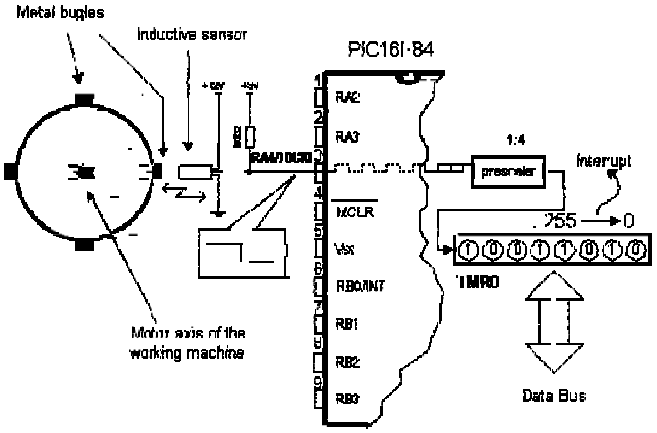
المقسم المبدئي Prescaler هو دارة داخلية في المتحكم يقوم بتقسيم النبضات قبل وصولها إلى المسجل Timer0. يتم تحديد الرقم المقسوم عليه من خلال الخانات الثلاث الأولى (PS2,PS1,PS0) في مسجل الخيار Option_Reg و أعلى رقم مقسوم عليه هو 256 مما يعني أنه مع كل 256 نبضة ساعة سيزداد عداد المؤقت بمقدار واحد مما يوفر إمكانية قياس فترات زمنية أطول. الخانة PSA في مسجل الخيار Option_Reg تحدد هل سيتم استخدام Prescaler للنبضات القادمة إلى Timer0 أم لا. عندما تأخذ صفر منطقي يتم استخدام Prescaler مع النبضات القادمة إلى المؤقت Timer0 . و عندما تأخذ واحد منطقي لا يتم استخدام Prescaler مع النبضات القادمة إلى للمؤقت Timer0.

هناك دارة مزامنة تجعل نبضات الساعة الخارجية متزامنة مع نبضات الساعة الداخلية وهذا ما يسبب تأخيراً مقداره دورتي آلة.

خلال الانتقال من 255 إلى الصفر فإن خانة TOIF في مسجل التحكم بالمقاطعات INTCON register تأخذ واحد منطقي لتدل على حدوث طفحان للمؤقت.

إن طرق الاستفادة من هذه العمليات و الخصائص ترجع للمبرمج حسب احتياجاته.

مثال : في الحياة العملية فإن أحد الأمثلة التي يمكن حلها عن طريق ساعة خارجية و المؤقت Timer0 هو حساب عدد دورات محرك الموضح بالشكل. بوضع أربعة مسامير على محاور اللفات فإن هذه المسامير تمثل المادة المعدنية و التي سوف تؤثر بالحساس. وبوضع حساس حث Inductive sensor على مسافة 5mm من رأس المسمار فإن حساس الحث ينتج إشارة هابطة في كل مرة يكون فيها رأس المسمار موازياً لرأس الحساس و تمثل كل إشارة ربع لفة كاملة . إن معدل المقياس يتم ضبطه بحيث يزداد المؤقت Timer0 بمقدار واحد مع كل أربع نبضات على القطب RA4 و بالتالي فإن الدورة الكاملة (أربع نبضات) سيزداد المؤقت Timer0 بمقدار واحد أي أننا سنحصل على عدد دورات المحرك مباشرة من المؤقت Timer0.



Determining a number of full axis turns of the motor

٨- مسجل الحالة STATUS REGISTER

عنوانه 03h و 83h (البنك 0 و البنك 1).

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	C
bit 7							bit 0

قيمة الخانة / الخانة	0	1
(IRP)7 دائماً قيمتها 0 في 16f84	يتم اختيار البنكين 0 و 1 في العنونة غير المباشرة	يتم اختيار البنكين 2 و 3 في العنونة غير المباشرة
(RP1)6 دائماً قيمتها 0 في 16f84	يتم اختيار البنكين 0 و 1 في العنونة المباشرة	يتم اختيار البنكين 2 و 3 في العنونة المباشرة
(RP0)5	يتم اختيار البنك 0 إذا كان PR1 = 0 يتم اختيار البنك ٢ إذا كان PR1 = 1	يتم اختيار البنك 1 إذا كان PR1 = 0 يتم اختيار البنك ٣ إذا كان PR1 = 1
(TO)4	عند انتهاء مدة مؤقت المراقبة WDT	عند بداية تشغيل المتحكم أو عند استخدام الأمر CLRWDT أو SLEEP
(PD)3	عند استخدام الأمر SLEEP	عند بداية تشغيل المتحكم أو عند استخدام الأمر CLRWDT
(Z)2 (علم الصفر)	عندما تكون نتيجة عملية ما لا تساوي الصفر	عندما تكون نتيجة عملية ما تساوي الصفر
(DC)1 (علم نصف الحمل)	عند عدم حدوث حمل في الخانة الرابعة	عند حدوث حمل في الخانة الرابعة
٠(C) علم الحمل	عند عدم حدوث حمل من الخانة الثامنة و الأخيرة	عند حدوث حمل من الخانة الثامنة و الأخيرة

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

- كيفية الانتقال إلى موقع ما في الذاكرة RAM (العنونة)

يمكن الوصول إلى أي موقع ذاكري في ذاكرة المعطيات بواسطة العنونة المباشرة أو العنونة غير المباشرة.

- العنونة المباشرة Direct Addressing

تتم العنونة المباشرة على مرحلتين :

١- اختيار البنك: هناك خانتان(RP1 , RP0) في مسجل الحالة status register – وهو من مسجلات الأغراض الخاصة – يتم من خلالهما اختيار البنك الهدف .

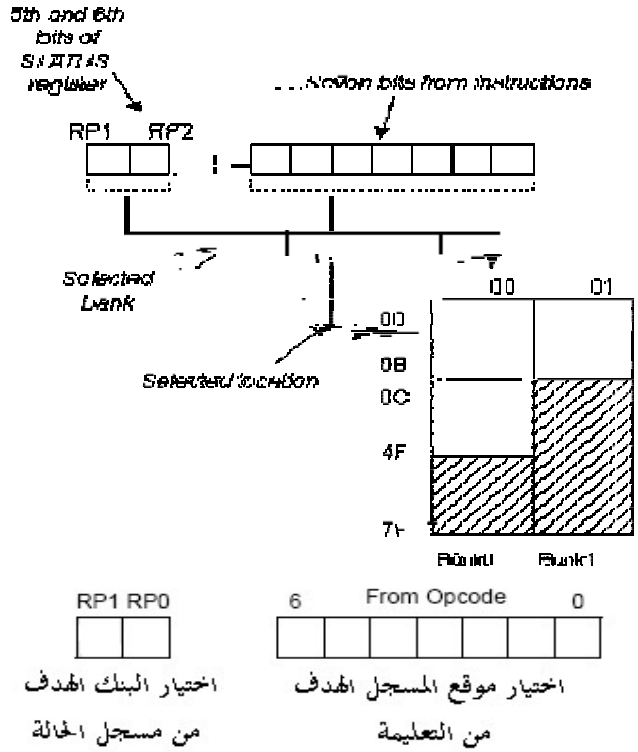
RP1	RP0	Bank
0	0	Bank0
0	1	Bank1
1	0	Bank2
1	1	Bank3

بالتالي RP1 = 0 دائماً في متحكم 16F84 لأنه يحتوي بنكين فقط . RP0 هي التي تحدد البنك الأول و الثاني.

٢- اختيار موقع ذاكري (مسجل) ضمن البنك المحدد مسبقاً . وبما أن البنك له سعة 128Byte بالتالي فهو يحتاج إلى 7 خطوط لعنونة مواقعه لأن $2^7 = 128$

إن التعليمة في 16F84 تمثل كلمة word (أي 14 bit) :

- 7 bit لاختيار الموقع الذاكري ضمن البنك المحدد .
- 7 bit لاختيار نوع العملية (نقل – جمع --------)



٢-٦- المذبذب Oscillator

تستخدم دائرة المذبذب لتزويد المتحكم بنبضات الساعة. يستطيع المتحكم 16f84A العمل مع أربع تكوينات مختلفة للمذبذبات وهي

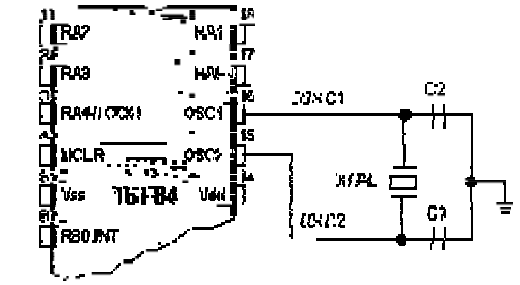
LP : Low Power Crystal

XT : Crystal / Resonator
HS : High Speed Crystal / Resonator
RC : Resistor / Capacitor

الثلاثة الأولى تعد ضمن تكوين مذبذب البلورة (كريستال) و الأخيرة تعد ضمن تكوين مذبذب المقاومة والمكثف.

٣-٦-١- مذبذب البلورة (كريستال)

يتم وضع مذبذب البلورة ضمن غلاف معدني مع إبرتين يتم وصلهما مع القطبين (OSC1 – OSC2) كما هو موضح بالشكل التالي:



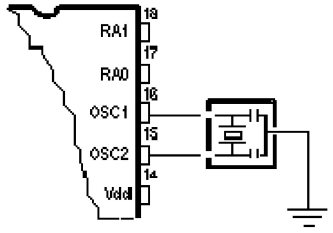
Connecting the quartz oscillator to give clock to a microcontroller

يكتب تردد البلورة عادة على غلافها الخارجي و هذا مهم لمعرفة تردد الاهتزاز أو تردد التشغيل الذي يعمل عليه المتحكم. والجدول التالي يوضح ترددات البلورات التي يمكن وصلها مع 16F84 و المكثفات السيراميكية المقابلة لكل تردد.

Mode	Freq	OSC1/C1	OSC2/C2
LP	32 kHz 200 kHz	68 - 100 pF 15 - 33 pF	68 - 100 pF 15 - 33 pF
XT	100 kHz 2 MHz 4 MHz	100 - 150 pF 15 - 33 pF 15 - 33 pF	100 - 150 pF 15 - 33 pF 15 - 33 pF
HS	4 MHz 20 MHz	15 - 33 pF 15 - 33 pF	15 - 33 pF 15 - 33 pF

ملاحظة: كلما ازدادت سعة المكثفة كلما ازداد الاستقرار ولكن كلما ازداد زمن بداية التشغيل. يمكن عدم وضع المكثفات ولكن ستقل استقرارية المتحكم .

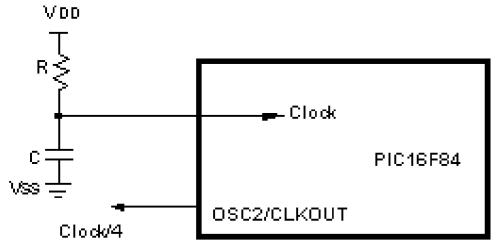
يمكن وضع المذبذب و المكثفات في حزمة واحدة بثلاثة إبر، ويسمى هذا العنصر مذبذب رنين سيراميكي Ceramic resonator. يوصل طرفه الأوسط إلى الأرضي بينما توصل الأطراف الأخرى مع المتحكم إلى الأطراف (OSC1 – OSC2).



Connecting a resonator onto a microcontroller

٢-٦-٢- مذبذب المقاومة المكثف RC Oscillator

في التطبيقات التي تكون فيها دقة الوقت غير هامة يوفر مذبذب المقاومة المكثف التكاليف. يعتمد تردد الرنين للمذبذب RC على معدل جهد التغذية و المقاومة R و سعة المكثفة C و درجة حرارة التشغيل. يوضح الشكل التالي كيفية توصيل مذبذب المقاومة المكثف مع المتحكم 16F84.

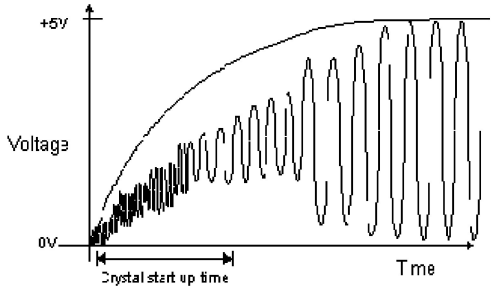


عندما تكون المقاومة أقل من 2.2K **Ω** لا يكون المذبذب مستقراً أو قد يتوقف عن العمل ، وبقيمة مقاومة عالية 1M **Ω** يصبح المذبذب حساساً للضوضاء و الرطوبة. والقيمة الموصى بها تتراوح مابين :

$$5\ k\Omega \le R_{EXT} \le 100\ k\Omega$$

يستخدم المكثف بقيم أعلى من 20pF للتخلص من الضوضاء ولزيادة الاستقرار.

المذبذبات تأخذ في البداية فترة عدم استقرار في التوقيت و المطال لكن بعد وقت قليل تستقر. ولمنع عدم دقة الساعة من التأثير على أداء المتحكم نحتاج وضع المتحكم في حالة تصفير خلال فترة عدم استقرار المتحكم وهذا ما يتم من خلال وجود مؤقتين داخل المتحكم : الأول Power up Timer (PWRT) الذي يجعل المتحكم في حالة تصفير Reset لمدة ثابتة هي 72 ms قبل بدء التشغيل. الثاني Oscillator Start up Timer(OST) الذي يجعل المتحكم بحالة Reset حتى يستقر مذبذب الكريستالة ،



Signal of an osc llatcr clcck after receiving the supply of a microcontroller

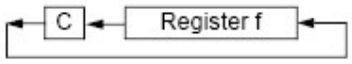
٢-٧- تعليمات لغة الاسمبلي

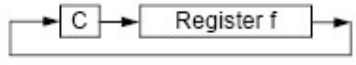
سنستخدم في تعليمات الاسمبلي الرموز التالية :

- الحرف f يشير إلى عنوان أحد مسجلات الذاكرة RAM.
- الحرف w يشير إلى المراكم (مسجل العمل) وهو عبارة عن مسجل يقوم المعالج بوضع القيمة الناتجة فيه.
- الحرف b يشير إلى رقم البت في مسجل ما.
- الحرف d يأخذ : 0 عندما نريد تخزين الناتج في المراكم.
  1 عندما نريد تخزين الناتج في المسجل f .
- الحرف k يشير إلى ثابت ما.
- PC يشير إلى عداد البرنامج .

شكل التعليمة	شرح التعليمة	تأثير التعليمة
BCF f,b	اجعل بت معين b الموجود في المسجل الذي عنوانه f يساوي 0	----
BSF f,b	اجعل بت معين b الموجود في المسجل الذي عنوانه f يساوي 1	----
MOVLW k	إسناد القيمة k للمراكم (W=k) .	----
MOVWF f	نقل القيمة المخزنة في المراكم إلى المسجل الذي عنوانه f .	----
MOVF f,d	نقل القيمة المخزنة في المسجل الذي عنوانه f إلى : المراكم : إذا كان d=0 المسجل نفسه f : إذا كان d=1	Z
BTFSC f,b	افحص بت معين b الموجود في المسجل الذي عنوانه f : إذا كان 1 : نفذ التعليمة التالية . إذا كان 0 : لا تنفذ التعليمة التالية .	----
BTFSS f,b	افحص بت معين b الموجود في المسجل f : إذا كان 1 : لا تنفذ التعليمة التالية . إذا كان 0 : نفذ التعليمة التالية .	----
DECF f,d	انقص واحد من المسجل الذي عنوانه f وخزن الناتج في: المراكم : إذا كان d=0 المسجل نفسه f : إذا كان d=1	Z

INCF f,d	زد واحد في المسجل الذي عنوانه f وخزن الناتج في : المراكم : إذا كان d=0 المسجل نفسه f : إذا كان d=1	Z
DECFSZ f,d	انقص واحد من المسجل الذي عنوانه f وخزن الناتج في: المراكم : إذا كان d=0 المسجل نفسه f : إذا كان d=1 إذا كان الناتج يساوي 0 فلا تنفذ التعليمة التالية . إذا كان الناتج لا يساوي 0 نفذ التعليمة التالية .	----
INCFSZ f,d	زد واحد في المسجل الذي عنوانه f وخزن الناتج في : المراكم : إذا كان d=0 المسجل نفسه f : إذا كان d=1 إذا كان الناتج يساوي 0 فلا تنفذ التعليمة التالية . إذا كان الناتج لا يساوي 0 نفذ التعليمة التالية .	---
COMF f,d	إتمام أحادي للمسجل f (قلب الأصفار والواحدات) و الناتج يخزن في : المراكم : إذا كان d=0 المسجل نفسه f : إذا كان d=1	Z
CLRF f	تصفير المسجل الذي عنوانه f .	Z
CLRW	تصفير المراكم w .	Z
CLRWDT	WDT=00h,WDT prescaler=0,TO=1,PD=1	TO,PD
SLEEP	WDT=00h,WDT prescaler=0,TO=1,PD=0	TO,PD
ANDLW k	إجراء عملية AND المنطقية (W and k) والناتج يخزن في المراكم حيث k ثابت من 0 وحتى 255.	Z
ANDWF f,d	إجراء عملية and المنطقية (W and f) والناتج يخزن في : المراكم : إذا كان d=0 المسجل نفسه f : إذا كان d=1	Z
IORLW k	إجراء عملية OR المنطقية (W or k) والناتج يخزن في المراكم.	Z

IORWF f,d	إجراء عملية OR المنطقية (W or f) والناتج يخزن في: المراكم : إذا كان d=0 المسجل نفسه f : إذا كان d=1	Z
XORLW k	إجراء عمليةXOR المنطقية (W xor k) والناتج يخزن في المراكم	Z
XORWF f,d	إجراء عملية XOR المنطقية (W xor f) والناتج يخزن في : المراكم : إذا كان d=0 المسجل نفسه f : إذا كان d=1	Z
ADDLW k	إجراء عملية الجمع (W + k) والناتج يخزن في المراكم .	C,DC,Z
ADDWF f,d	إجراء عملية الجمع (W + f) والناتج يخزن في : المراكم : إذا كان d=0 المسجل نفسه f : إذا كان d=1	C,DC,Z
SUBLW k	إجراء عملية الطرح (W – k) والناتج يخزن في المراكم .	C,DC,Z
SUBWF f,d	إجراء عملية الطرح (W– f) والناتج يخزن في : المراكم : إذا كان d=0 المسجل نفسه f : إذا كان d=1	C,DC,Z
RLF f,d	تحريك المسجل الذي عنوانه f خانة واحدة باتجاه اليسار ونقل قيمة الحمل C إلى الخانة الأقل أهمية في المسجل ونقل الخانة الأكثر أهمية إلى الحمل C والناتج يخزن في : المراكم : إذا كان d=0 المسجل نفسه f : إذا كان d=1 	C
RRF f,d	تحريك المسجل الذي عنوانه f خانة واحدة باتجاه اليمين ونقل قيمة الحمل C إلى الخانة الأكثر أهمية في المسجل ونقل الخانة الأقل أهمية إلى الحمل C والناتج يخزن في : المراكم : إذا كان d=0 المسجل نفسه f : إذا كان d=1	C

		
SWAPF f,d	نقل البتات من 0 وحتى 3 في المسجل الذي عنوانه f إلى المواقع من 4 وحتى 7 في المراكم : إذا كان d=0 المسجل نفسه f : إذا كان d=1 نقل البتات من 4 وحتى 7 في المسجل الذي عنوانه f إلى المواقع من 0 وحتى 3 في المراكم : إذا كان d=0 المسجل نفسه f : إذا كان d=1	----
CALL k	TOS=PC+1 , PC<0:10>=K حيث k ثابت من 0 وحتى 2047.	----
GOTO k	PC<0:10>=K حيث k ثابت من 0 وحتى 2047.	----
RETURN	PC=TOS	----
RETLW k	PC=TOS , W=K	----
RETFIE	PC=TOS , GIE=1	----
NOP	لا تنفذ شيء	----

برمجة المتحكم باستخدام لغة الاسمبلي

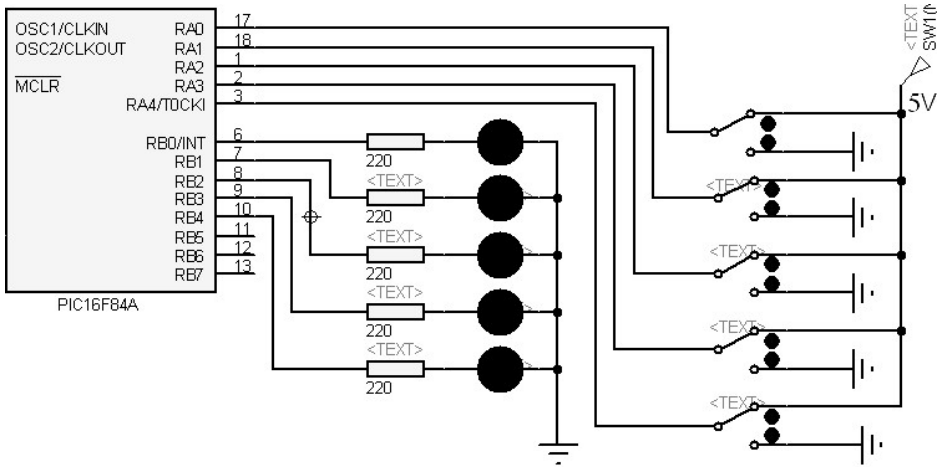
٣-١- الشكل العام للبرنامج بلغة الاسمبلي

```
#include <اسم المتحكم.h>                    // استدعاء لمكتبة المتحكم
#use delay(clock=تردد الكريستالة)           // إدخال تردد الكريستالة
                                            // تعريف لعناوين مسجلات الذاكرة RAM المعروفة
#define STATUS  3
#define TRISA   5
#define TRISB   6
#define PORTA   5
#define PORTB   6
#define OPTION_REG   1
#define INTCON   11
#define RP0  5

void main()
{
#asm

جسم البرنامج

#endasm
}
```

٣-٢- تطبيقات عملية بسيطة بلغة الاسمبلي

١- قراءة أرجل البوابة A و إخراج النتيجة على البوابة B

في هذا التطبيق يتم إدخال القيم الرقمية المطبقة على أرجل البوابة A (A0,A1,A2,A3, A4) و إخراج هذه القيم على البوابة B (B0,B1,B2,B3, B4) و على الترتيب.



خطوات كتابة البرنامج

١- تفعيل النافذة A على أنها نافذة دخل من خلال وضع واحدات في المسجل TRISA ، تفعيل النافذة B على أنها نافذة خرج من خلال وضع أصفار في المسجل TRISB .

٢- نقل القيمة الموجودة في المسجل PORTA – هي القيمة المطبقة على أرجل البوابة A- إلى مسجل العمل W.

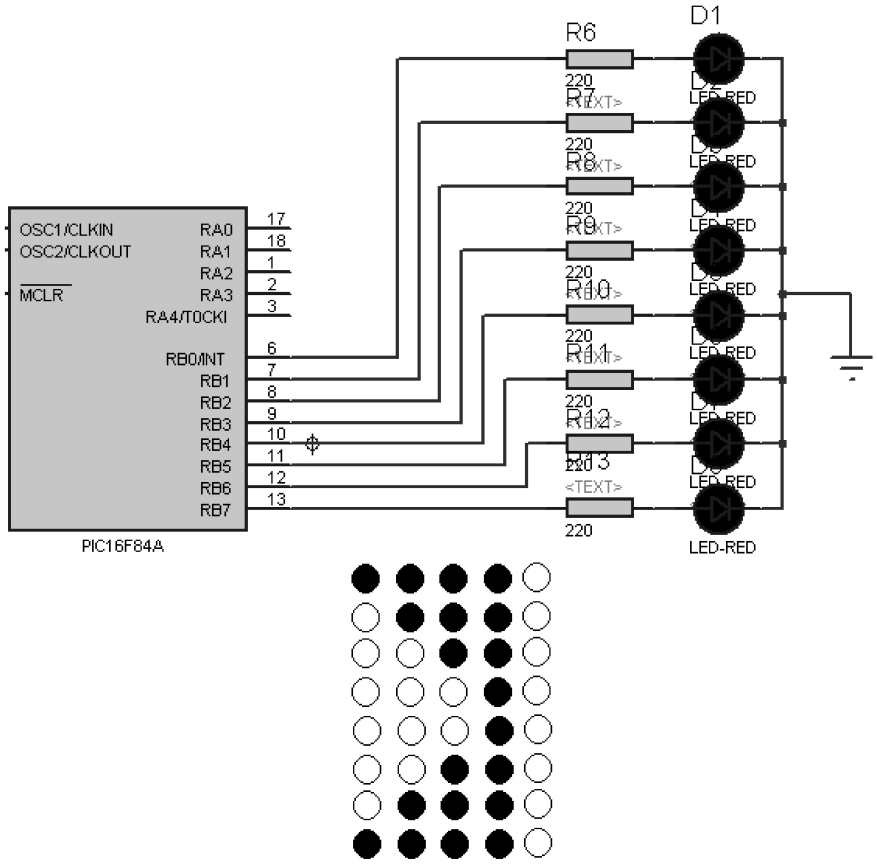
٣- نقل القيمة الموجودة في مسجل العمل W إلى المسجل PORTB لإظهار تلك القيمة على النافذة B.

٤- تكرار للخطوتين 2 و ٣ لكي يستمر البرنامج بقراءة النافذة A و إخراج القيمة على النافذة B.

```
#include <16f84A.h>
#use delay(clock=4M)
#define STATUS  3
#define TRISA   5
#define TRISB   6
#define PORTA   5
#define PORTB   6
#define OPTION_REG   1
#define INTCON   11
#define RP0  5

void main()
{
#asm
  BSF    STATUS.RP0                  //RP0=1 , BANK1
  MOVLW  0x1F                        // 0x1F →    W
  MOVWF  TRISA                       //   W   →  TRISA
  MOVLW  0x00                        //  0x00 →    W
  MOVWF  TRISB                       //   W   →  TRISB

  BCF    STATUS.RP0                   //RP0=0 , BANK0
  LOOP:
  MOVF   PORTA,W                     // PORTA →    W
  MOVWF  PORTB                       //   W     → PORTB
  GOTO   LOOP
#endasm
}
```

٢- الثنائيات الضوئية

في هذا التطبيق سنقوم بوصل ثمانية ثنائيات ضوئية على النافذة B. سنضيء الثنائيين الطرفيين بداية ثم الثنائيين اللذين بجوارهما و هكذا حتى تضيء كل الثنائيات ونعاود ذلك من جديد كما هو مبين بالشكل التالي من اليسار لليمين ، بين كل مرحلة و أخرى هناك تأخير زمني لكي تلاحظ العين ذلك.



خطوات كتابة البرنامج

١- تفعيل النافذة B على أنها نافذة خرج من خلال وضع أصفار في المسجل TRISB .

٢- نقل قيمة الحركة المطلوبة K إلى مسجل العمل W

٣- نقل القيمة الموجودة في مسجل العمل W إلى المسجل PORTB لإظهار تلك القيمة على النافذة B.

٤- - تكرار للخطوتين 2 و ٣ مع تغيير قيمة الحركة K كل مرة

```
#include <16f84A.h>
#use delay(clock=4M)

#define STATUS  3
#define TRISA   5
#define TRISB   6
#define PORTA   5
#define PORTB   6
#define OPTION_REG   1
#define INTCON   11
#define RP0  5
#define COUNT1 0X0C
#define COUNT2 0X0E
```

```
void main()
{
#asm
  BSF    STATUS,RP0          // RP0=1 , BANK1
  MOVLW  0x00                // 0x00 → W
  MOVWF  TRISB               // W → TRISB

  BCF    STATUS,RP0          // RP0=0 , BANK0

UP:
  MOVLW  0b10000001
  MOVWF  PORTB
  CALL   DELAY
  MOVLW  0b11000011
  MOVWF  PORTB
  CALL   DELAY
  MOVLW  0b11100111
  MOVWF  PORTB
  CALL   DELAY
  MOVLW  0b11111111
  MOVWF  PORTB
  CALL   DELAY
  MOVLW  0b00000000
  MOVWF  PORTB
  CALL   DELAY
  GOTO   UP

DELAY:
  MOVLW  0xFF
  MOVWF  COUNT1
LOOP1:
  MOVLW  0xFF
  MOVWF  COUNT2
LOOP2:
  DECFSZ COUNT2,F
  GOTO   LOOP2
  DECFSZ COUNT1,F
  GOTO   LOOP1
  RETURN
#endasm
}
```

برمجة المتحكم باستخدام لغة C Compiler

٤-١- هيكلة البرنامج :

```
#include < اسم المتحكم .h>
#use delay(clock=تردد الكريستالة)
#include < اسم مكتبة ما.h >
التصريح عن متغيرات عامة ( تظهر في البرنامج الرئيسي و الفرعي )

Void  اسم البرنامج الفرعي ( متغيرات يتم تمريرها إلى البرنامج الفرعي )    // إجرائية البرنامج الفرعي
{
التصريح عن متغيرات تظهر في البرنامج الفرعي فقط
        جسم البرنامج الفرعي
}

Void main( )                                                    // البرنامج الرئيسي
{
التصريح عن متغيرات تظهر في البرنامج الرئيسي فقط
        جسم البرنامج الرئيسي
}
```

٤-٢- أهم المكتبات القياسية التي يتم استدعاؤها

- #include <math.h>

يتم استدعاء هذه المكتبة عند الحاجة إلى استخدام بعض التعليمات الرياضية مثل Sin ( ) , log ( ) , floor ()

- #include <string.h>

يتم استدعاء هذه المكتبة عند استخدام بعض التعليمات التي تتعامل مع السلاسل المحرفية.

- #include <stdlib.h>

يتم استدعاء هذه المكتبة عند استخدام بعض التعليمات مثل Atoi( ) .....

٤-٣- التصريح عن المتغيرات

الشكل العام للتصريح عن المتغيرات في لغة C هو :

نمط المتغير اسم المتغير ;

الأنماط الأساسية التي يمكن تعريف المتغيرات من خلالها يمكن إيضاحها بالجدول التالي :

نمط (نوع) المتغير	الأنماط المكافئة	السعة	المجال	
			بدون إشارة Unsigned	مع إشارة Signed
int1	short , Boolean	1 bit	عدد صحيح 0 to 1	N/A
int8	int , Byte	8 bit	عدد صحيح 0 to 255	عدد صحيح -128 to 127
int16	long	16 bit	عدد صحيح 0 to 65535	عدد صحيح -32768 to 32767
int32	long long	32 bit	عدد صحيح 0 to 4294967295	عدد صحيح -2147483648 to 2147483647
float32	float	32 bit	عدد حقيقي -1.5×10^{45} to 3.4×10^{38}	
char		8 bit	محرف يرمز برموز أسكي وله مجال 0 to 255	

الأنماط السابقة هي بشكل افتراضي بدون إشارة unsigned ما عدا float32. و لكي تصبح بإشارة فقط عندما نضع قبل النمط عبارة signed.

أمثلة

- **Int1 x;** : يتم حجز بت واحد من الذاكرة RAM اسمه x. يأخذ x إما 0 أو 1.
- **short x;** : يتم حجز بت واحد من الذاكرة RAM اسمه x. (تكافئ العبارة السابقة).
- **Int8 x;** : يتم حجز بايت واحد من الذاكرة RAM اسمه x. مجال x من 0 و حتى 255 بأعداد صحيحة.
- **Signed int8 x;** : يتم حجز بايت واحد من الذاكرة RAM اسمه x. مجال x من 128- و حتى 127 بأعداد صحيحة.
- **float x;** : يتم حجز 32 bit أي 4 بايت من الذاكرة RAM اسمه x. مجال x من الجدول السابق.

٤-٤- الحلقات : تهدف إلى تكرار تنفيذ مجموعة من الأوامر

أ- الحلقات غير الشرطية :

- for (x = قيمة ابتدائية ; x<= قيمة نهائية ; x = x + قفزة)

{ أوامر }

- for (;;) // تنفيذ لا نهائي

{ أوامر }

ب- الحلقات الشرطية :

- while (شرط منطقي)

{ أوامر }

- while (true أو 1) // تنفيذ لا نهائي

{ أوامر }

يتم كسر الحلقة (أي الخروج منها) من خلال تعليمة break.

٤-٥- استخدام تعليمة IF الشرطية : تهدف إلى تقييد تنفيذ أوامر بتحقق شرط معين

If (شرط منطقي)

{ أوامر }

If (شرط منطقي)

{ أوامر }

Else

{ أوامر }

ما هو الشرط المنطقي ؟

هو عملية منطقية تكون نتيجتها True أو False ، فعندما تكون النتيجة True (1) يتم تنفيذ أوامر while أو if.....، أما False (0) فلا يتم تنفيذها . و أهم العمليات المنطقية :

A== ٣ , A!=٣ , A<٣ , A>3 , A<=٣ A>=3

من الممكن أيضاً دمج عمليتين منطقيتين باستخدام AND : A==3 && B!=8

من الممكن أيضاً دمج عمليتين منطقيتين باستخدام OR : A==3 || B!=8

٤-٦- عمليات الإدخال و الإخراج الرقمية على البوابات :

١ - عملية الإخراج :

أ - عملية الإخراج على بوابة كاملة :

Output_اسم البوابة(value);

Output_a(value); Output_b(value); ……