الفصل الرابع ٤

المتحكمات بيك متوسطة الإمكانيات

The Midrange PIC Microcontrollers

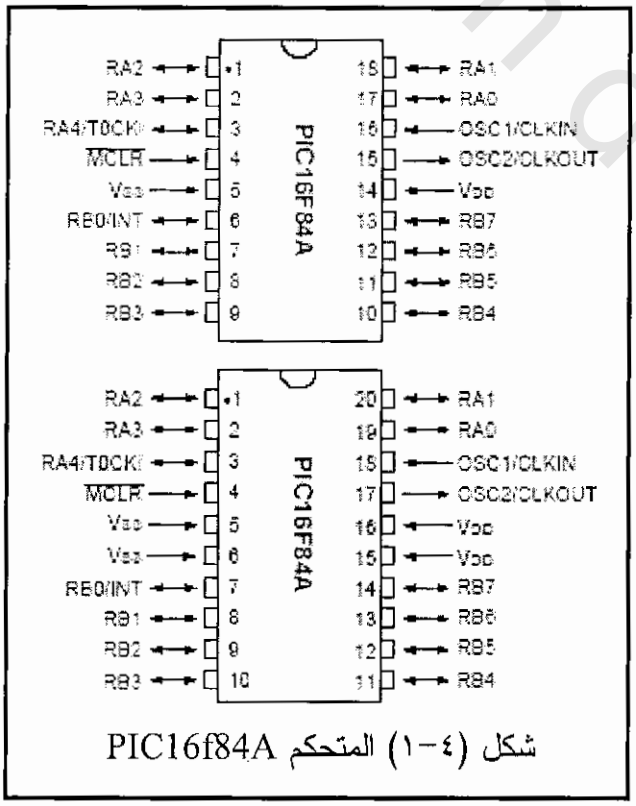
٤-١ مقدمة

بالرغم من وجود العديد من الشركات التى تنتج المتحكمات المختلفة ، إلا أن عائلة المتحكمات بيك PIC التى تنتجها شركة Microchip تعتبر الأكثر شيوعا من حيث الاستخدام نتيجة تواجدها فى أكثر من إصدار تناسب العديد من التطبيقات . ونحن هنا سنقدم شرحا تفصيليا لمتحكمات هذه العائلة ذات الإمكانيات المتوسطة ممثلة فى المتحكم PIC16f84 نظرا لملائمتها أيضا للكثير من التطبيقات . ولقد قدمنا فى الفصل السابق شرحا للتركيب الداخلى للمتحكمات بسيطة الإمكانيات ممثلة فى المتحكم PIC12c508 وأرجأنا الحديث عن مجموعة أوامره حيث سيتم شرحها فى هذا الفصل نظرا لتطابق جموعة الأوامر فى المعالجين .

٤-٢ الخواص العامة لهذا المتحكم

شكل (٤-١) يبين بعض شرائح هذا الإصدار التى تشترك كلها فى الخواص العامة التالية :

- شرائح ذات ١٨ أو ٢٠ طرف ، لاحظ فى شكل (٤-١) أن الفرق هو فقط تكرار أطراف القدرة .
- عدد ٣٥ أمر فقط ، كل منها يتكون من ١٤ بت وكلها تنفذ فى دورة ماكينة machine cycle واحدة .
- 1024x12 مكان ذاكرة برمجة ، لاحظ أن عدد بتات مكان البرمجة هو



شكل (٤-١) المتحكم PIC16f84A

نفسه عدد بتات الأمر ، ١٤ بت .
- عدد ١٥ مسجلا ذات أغراض خاصة ، سنعرف وظيفة كل منها بعد قليل .
- مكدسة stack من ٨ مستويات فقط .
- مسار بيانات من ٨ بت لنقل البيانات بين المسجلات ووحدة الحساب وذاكرة البيانات .
- نبضات تزامن من صفر حتى ٢٠ ميجاهرتز .
- ٦٨ بايت ذاكرة بيانات RAM .
- ٦٤ بايت ذاكرة بيانات EEPROM .
- ١٣ خط إدخال/إخراج للبيانات تستطيع أن تغذى أو تبتلع حتى ٢٥ ميللى أمبير يمكنها أن إدارة LEDS أو ما يعادلها .
- مؤقت ٨ بت مع عداد قاسم يستخدم لإطالة زمن التوقيت .

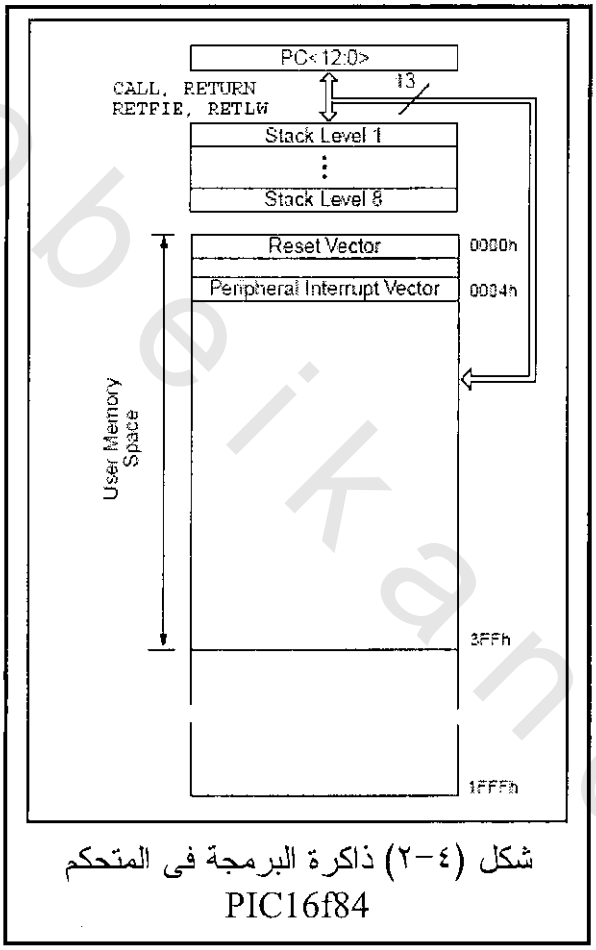
٤-٣ وظيفة الأطراف المختلفة فى المتحكم PIC16f84

- **الطرف ١٦ OSC1/CLKIN** : يوصل عليه أحد طرفى الكريستال الخارجية ، كما يتم إدخال نبضات التزامن من أى مصدر خارجى على هذا الطرف .
- **الطرف ١٥ OSC2/CLKOUT** : يوصل عليه الطرف الثانى من الكريستال الخارجية ، وفى هذه الحالة تؤخذ من عليه نبضات التزامن . فى حالة استخدام مذبذب داخلى RC يؤخذ من على هذا الطرف نبضات تزامن تمثل دورة الأمر machine cycle والتى تساوى ربع نبضات التزامن Fosc/4 .
- **الطرف ٤** $\overline{MCLR}$: طرف تصفير منخفض الفعالية بوضعه بصفر يتم تصفير عداد البرنامج . يعتبر طرف قدرة فى حالة كتابة البرنامج فى المتحكم .
- **الأطراف ١٧ و ١٨ و ١ و ٢ و ٣** وهى تمثل جميعها أطراف إدخال وإخراج Ra0 و Ra1 و Ra2 و Ra3 و Ra4/T0CKL . فقط الطرف Ra4 يستخدم لإدخال نبضات تزامن خارجية للمؤقت T0 .
- **الأطراف ٦ حتى ١٣** تمثل جميعها أطراف إدخال وإخراج Rb0/INT و Rb1 حتى Rb7 . كما نلاحظ فإن الخط Rb0/INT يستخدم أيضا كطرف مقاطعة خارجية للمتحكم . الطرف RB6 يستخدم لإدخال نبضات التزامن فى حالة برمجة الشريحة تتابعيا ، والطرف RB7 يستخدم لإدخال بيانات البرنامج نفسه .
- **الطرف ٥** Vss أو الأرضى .
- **الطرف ١٤** Vdd أو طرف القدرة من ٢ حتى ٥,٥ فولت .

٤-٤ ذاكرة البرمجة فى المتحكم PIC16f84

ذاكرة البرمجة فى هذا النوع من المتحكمات تتكون من ١٠٢٤×١٤ بايت (صفحتان من العنوان 0000h حتى العنوان 03FFh) ، أى أن البايت هنا تتكون من ١٤ بت إذا

جاز لنا أن نطلق عليها بايت . أى أن جميع أوامر هذا الإصدار من بيك تتكون من ١٤ بت . عداد البرنامج PC يتكون من ١٣ بت ، أى أنه يمكنه التعامل مع ذاكرة مقدارها ٨ كيلوبايت×١٤ بت ، ولكن الموجود فقط داخل الشريحة هو ١٠٢٤ بايت فقط . عداد البرنامج يحتوى عنوان الأمر التالى فى التنفيذ كما أشرنا سابقا . شكل (٤-٢) يبين رسما صندوقيا لذاكرة البرمجة فى هذه المتحكمات .



شكل (٤-٢) ذاكرة البرمجة فى المتحكم PIC16f84

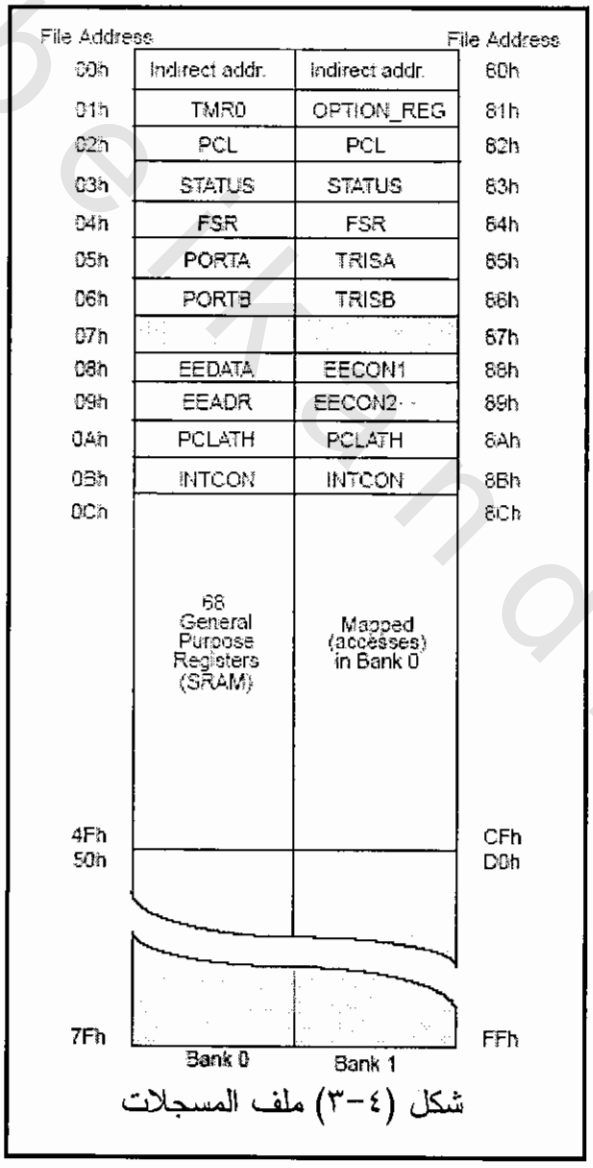
عند عمل تصفير للمتحكم فإنه يذهب للعنوان 0000h ، والعنوان 0004h هو عنوان متجه المقاطعة .

نلاحظ من شكل (٤-٢) أن المكدسة stack تتكون من ٨ أماكن فقط ، عرض كل منهما أيضا يتكون من ١٣ بت . يوجد هنا مؤشر للمكدسة SP ولكنه غير معنون ولا يتم التعامل معه بأى أمر . مؤشر المكدسة يشير على قمة المكدسة ومع كل إضافة فى المكدسة يتم زيادة المؤشر بواحد ، ومع كل سحب من المكدسة ينقص المؤشر بمقدار واحد . إذا وصل المؤشر لآخر مكان فى المكدسة ، فإنه مع عملية الإضافة التالية يشير المؤشر إلى أول مكان مرة أخرى حيث تتم الإضافة الجديدة هناك .

٤-٥ ذاكرة البيانات RAM

كما ذكرنا سابقا فإن ذاكرة البرمجة فى كل متحكمات PIC يتم التعامل معها من خلال مسار بيانات data bus مختلف تماما عن مسار البيانات الذى يستخدم فى التعامل مع ذاكرة البيانات . فى المتحكم 16f84 يتم التعامل مع ذاكرة برمجة من خلال مسار بيانات ١٤ بت ، والتعامل مع ذاكرة البيانات يكون من خلال مسار بيانات ٨ بت . يوجد نوعين من ذاكرة البيانات فى المتحكم 16f84 . النوع الأول ويتكون من ٦٨ بايت من ذاكرة القراءة والكتابة الاستاتيكية static RAM وهذه

سنتحدث عنها بالتفصيل فى الجزء القادم ويطلق عليها اسم ملف المسجلات register fie . النوع الآخر من ذاكرة البيانات يتكون من ٦٤ بايت من الـذاكرة EEPROM وهى ذاكرة قراءة فقط يمكن تخزين البيانات عليها بطرق إلكترونية ومسحها أيضـا بنفس الطريقة . هذه الذاكرة ذاكرة بيانات يتم التعامل معها من خلال نفـس مسـار البيانات (٨ بت) الخاص بالذاكرة الاستاتيكية سوى أن البيانـات عليهـا لا تضـيع بانقطاع القدرة ، لذلك فهى تستخدم لتخزين البيانات المستديمة مثل الجداول . البايت فى ذاكرة البيانات تتكون من ٨ بت .



شكل (٤-٣) ملف المسجلات

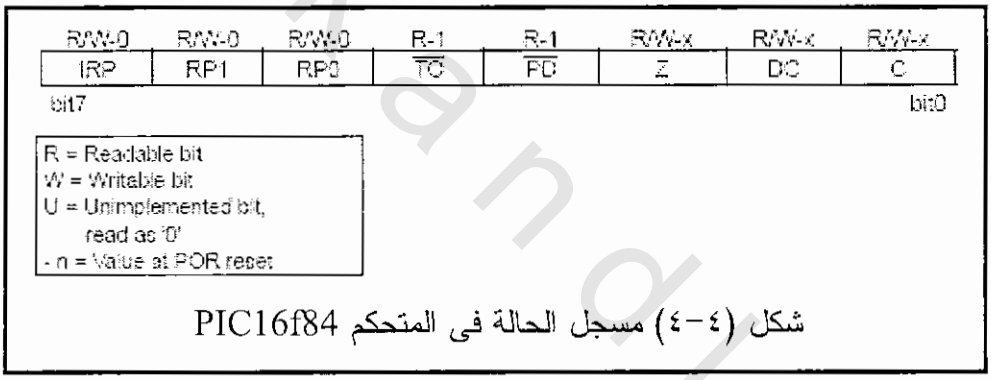
ملف المسجلات أيضا ينقسم إلى قسمين ، الأول عبارة عن مجموعة من المسجلات الخاصة Special Function Registers, SFR التى تؤدى وظيفة معينة خاصة بأداء المتحكم وسنرى وظيفة كل واحدة من هذه المسجلات فى الجزء القادم . الجزء الثانى من ملف المسجلات عبارة عن مجموعة من المسجلات العامة التى تستخدم فى كل أغراض البرمجة من خلال عنوانها مباشرة ، أو بطريقة غير مباشرة باستخدام المسجل FSR كما سنرى فيما بعد . ملف المسجلات فى المتحكم 16f84 يوجد فى بانكين ، البانك 0 ، والبانك 1 ، ويتم اختيار البانك كما سنرى من خلال علم فى مسجل الحالة . شكل (٤-٣) يبين رسما توضيحيا لمحتويات ملف المسجلات فى المتحكم 16f84 . المسجلات العامة الموجودة فى بانك 1 يتم التعامل معها من خلال نفس عناوين البانك 0 كما هو مبين فى شكل

(٤-٣) ، أى أن التعامل مع العنوان 8c فى البانك ١سيكون فى نفس الوقت تعامل مع العنوان 0c فى البانك 0 .
سنقدم فيما يلى شرحا مفصلا لوظيفة كل واحدة من المسجلات الخاصة والتى تشغل العناوين من 00h حتى 0Bh . بعض هذه المسجلات موجودة فى البانكين فى مكانين متقابلين تماما والتعامل مع أى عنوان منهما يكون تعاملا مع نفس المسجل ، وبعض هذه المسجلات غير مكرر .

٤-٦ المسجلات الخاصة فى المتحكم PIC16f84A

٤-٦-١ مسجل الحالة Status register, STATUS

هذا المسجل عنوانه هو 03h وهو يحتوى مجموعة أعلام flags حسابية تبين حالة المتحكم بعد كل عملية حسابية بالإضافة إلى بعض الأعلام الأخرى التى سيأتى شرحها الآن كما هو موضح فى شكل (٤-٤) . هذا الشكل يبين إذا كانت كل بت أو علم يمكن قراءتها أو الكتابة فيها وحالة كل علم من الأعلام عند عمل تصفير reset للمتحكم :



شكل (٤-٤) مسجل الحالة فى المتحكم PIC16f84

- **العلم IRP** غير مستخدم فى المتحكم 16f84 وهو يستخدم لاختيار بانك معين فى ملف المسجلات فى حالة العنونة الغير مباشرة .
- **العلم RP1, RP0** يستخدمان لاختيار أحد بانكات ملف المسجلات . RP1 غير مستخدم فى المتحكم 16f84A . عندما RP1:RP0=00 يتم اختيار بانك صفر ، وعندما RP1:RP0=01 يتم اختيار بانك واحد . كما أشرنا كل بانك يمتد حتى العنوان 7FFh ولكن الموجود منهم فقط هو حتى العنوان 4FFh .
- **العلم Time Out, $\overline{TO}$** وهو علم الأنتهاء من زمن تأخير ، حيث يكون واحد عند رجوع القدرة ، أو تنفيذ الأمر sleep للدخول فى النوم ، أو تنشيط أو بداية تشغيل مؤقت كلب الحراسة WDT . يكون صفر (نشط) عند تصفير مؤقت كلب الحراسة أى نفاذ زمن التأخير المخصص له .

- **العلم Power Down, $\overline{PD}$** وهو علم بيان حالة القدرة على الشريحة حيث يكون واحد عند استقرار القدرة أو تصفير مؤقت كلب الحراسة ، ويكون صفر فى حالة تنفيذ الأمر sleep .
- **العلم Zero flag, Z** ، هذا العلم يكون واحد إذا كانت آخر عملية حسابية أو منطقية نفذها المتحكم تساوى صفر ، ويكون صفر فيما عدا ذلك .
- **العلم Digit Carry, DC** ، وهو علم الحمل النصفى Half carry flag ، وهذا العلم يكون واحد إذا كان هناك حمل من البت الثالثة للرابعة بعد إجراء عملية جمع ، أو استلاف من البت الرابعة للثالثة فى حالة إجراء عملية طرح . أى أن عملية الحمل أو الاستلاف هنا تكون بين الخانة الأولى والثانية عند التعامل مع الأرقام الستعشرية . يكون صفر فيما عدا ذلك .
- **العلم Carry, C** ، وهو علم الحمل Carry flag الذى يكون واحد عند حدوث حمل من آخر بت (السابعة) فى حالة إجراء عملية جمع ، أو استلاف إليها فى حالة إجراء عملية طرح . يكون صفر فيما عدا ذلك .

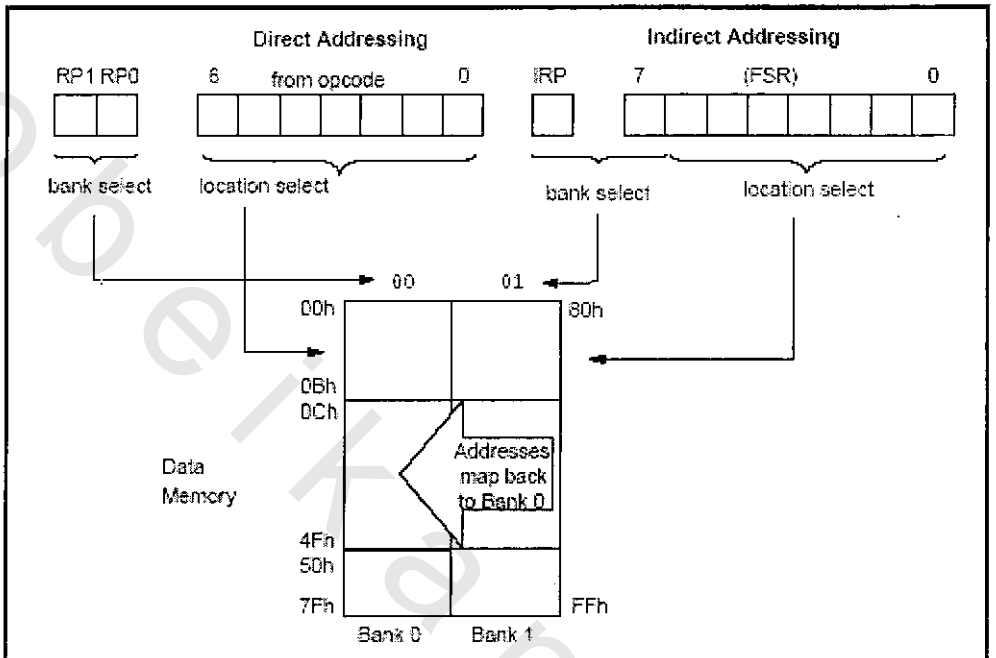
٤-٦-٢ مسجل ماسك عداد البرنامج Program Counter Latch, PCL

عداد البرنامج PC كالعادة يحتوى عنوان الأمر الذى عليه الدور فى التنفيذ ، وهو يتكون من ١٣ بت وبذلك يمكنه التعامل مع كمية من ذاكرة البرمجة تصل إلى ٨ كيلوبايت . تذكر أن المتحكم الذى معنا الآن يحتوى ١٠٢٤ بايت (صفحتان) من ذاكرة البرمجة . كما نعلم أيضا فإن كل أوامر البيك تتكون من بايت واحدة ، لذلك فإنه مع تنفيذ كل أمر فإن عداد البرنامج يزداد بمقدار واحد فقط إلا مع الأوامر التى تغير من محتويات هذا العداد مثل أوامر القفز أو النداء على البرامج الفرعية . البايت الأولى منه هى PCL وهى تتكون من ٨ بت ويمكن التعامل المباشر معها من خلال أوامر المتحكم . البايت الأعلى من عداد البرنامج هى PCH وهى ٥ بت فقط من البت ٨ حتى البت ١٢ ولا يمكن التعامل المباشر معها إلا من خلال المسجل PCLAH الذى عنوانه 0Ah فى ملف المسجلات كما فى شكل (٤-٣) .

٤-٦-٣ المسجلان FSR, INDF

المسجل File Select Register, FSR الذى عنوانه 04h فى ملف المسجلات يستخدم مع المسجل INDF للتعامل الغير مباشر مع المسجلات العامة . شكل (٤-٥) يبين طريقة التعامل المباشر وغير المباشر مع ملف المسجلات . فى الطريقة المباشرة يكون عنوان المسجل موجود مباشرة فى الأمر نفسه . أما فى الطريقة الغير مباشرة فإن عنوان المسجل المراد التعامل معه يوضع فى المسجل FSR ويستخدم المسجل INDF فى الأمر للدلالة على أن هذا التعامل من النوع الغير مباشر . فمثلا الأمر CLRF INDF معناه تصفير المسجل الذى يوجد عنوانه فى المسجل FSR . لاحظ أن المسجل INDF مسجل غير موجود أصلا داخل الشريحة وليس له عنوان ،

ومحاولة القراءة منه تعطى أصفارا ، ومحاولة الكتابة فيه لا تسجل أى نتيجة أى كما لو نفذنا أمر لا يعمل شيء ، وهو موجود فقط للتعامل الغير مباشر مع ملف المسجلات بالتعاون مع المسجل FSR كما سنرى .



شكل (٤-٥) التعامل المباشر وغير المباشر مع ملف المسجلات

مثال ٤-١

تصفير المسجلات التى عناوينها 20h حتى 2Fh بالطريقة الغير مباشرة .

```
        MOVLW  0x20
        MOVWF  FSR
Next    CLRF   INDF
        INCF   FSR,F
        BTFSC  FSR,4
        GOTO   Next
        Continue
```

الأمر الأول يضع الثابت أو القيمة الفورية 20h فى المسجل w ، ثم ينقل هذه القيمة من المسجل w إلى المسجل fsr كما فى الأمر الثانى وبذلك أصبح المسجل fsr يشير على العنوان 20h . الأمر الثالث clrf indf يصفر بطريقة غير مباشرة المسجل الذى عنوانه فى المسجل fsr . الأمر الرابع يزيد محتويات المسجل fsr بمقدار واحد لتشير إلى العنوان التالى . الأمر الخامس bitt test and skip if clear, BTFSC يختبر البت

الرابعة ليرى إذا كانت صفر يهمل الأمر التالى ولن ينفذ الأمر GOTO حيث عندها تكون محتويات المسجل fsr تساوى 30h ويستمر فى باقى البرنامج . أما إذا كانت البت الرابعة تساوى صفر فإنه ينفذ الأمر goto ويرجع إلى العلامة Next حيث يستمر فى تنفيذ الحلقة .

٤-٦-٤ إدخال وإخراج البيانات

إدخال وإخراج البيانات من أو إلى المتحكم يتم عن طريق البوابتين A و B . البوابة A تتكون من ٥ خطوط يمكن برمجتها كخطوط إدخال أو إخراج . البوابة B تتكون من ٨ خطوط أيضا يمكن برمجتها إدخال أو إخراج . كل واحدة من هذه البوابات لها مسجل فصل TRIS له نفس العدد من البتات ، أى أن هناك المسجلين TRISA وعنوانه 85h فى البانك 1 و TRISB وعنوانه 86h فى البانك 1 أيضا . بكتابة واحد فى أى بت من بتات هذين المسجلين TRISA أو TRISB فإن طرف البوابة المقابل له سيكون طرف دخل ، وبكتابة صفر فى أى بت من بتات المسجل TRISA أو TRISB فإن طرف البوابة المقابل سيكون طرف خرج .
البرنامج التالى يجعل PA0 حتى PA3 أطراف دخل والطرف PA4 طرف خرج :

```
BCF     STATUS,RP0
CLRF    PORTA
BSF     STATUS,RP0
MOVLW   0x0F
MOVWF   TRISA
```

لاحظ أنه عند استخدام أحد هذه الأطراف لوظيفة أخرى غير الإدخال والإخراج فإن مثل هذا الطرف لا يمكن استخدامه كطرف إدخال أو إخراج . فمثلا الطرف RA4 يمكن استخدامه لإدخال نبضات تزامن المؤقت T0 وفى هذه الحالة فإنه لا يستخدم كطرف إدخال أو إخراج .
البوابة B تعمل بنفس الطريقة مثل البوابة A سوى أنها مكونة من ٨ بت كلها يمكن برمجتها لتعمل خطوط إدخال أو إخراج باستخدام المسجل TRISB المكون من ٨ بت أيضا حيث بوضع واحد فى أى بت من بتات هذا المسجل فإن الطرف القابل له فى البوابة B سيكون طرف إدخال ، وبوضع صفر فى أى بت من بتات المسجل TRISB فإن الطرف المقابل له فى البوابة B سيكون طرف خرج .
الطرف RB0 فى البوابة B يعمل أيضا كطرف مقاطعة خارجية للمتحكم ، وعند استعماله لهذا الغرض فإنه لا يعمل كطرف إدخال وإخراج للبيانات .
أطراف البوابة RB4,5,6,7 كلها يمكن استخدامها لعمل مقاطعة توقظ المتحكم من حالة النوم التى يدخل فيها بالأمر sleep . فإذا كانت هذه الأطراف موصلة على لوحة مفاتيح مثلا فإن ضرب أى حرف من هذه اللوحة سيوقظ المتحكم .

٤-٦-٥ المسجل TMR0 أو المؤقت T0 وملحقاته

تتم عملية التوقيت فى هذا المتحكم من خلال المسجل TMR0 الذى يتكون من ٨ بت والذى يستخدم كمؤقت أو عداد كما سنرى . هناك أيضا عداد من ٨ بت يستخدم لضبط أو قاسم لساعة المؤقت prescaler ، هناك أيضا المسجل Option الذى يستخدم فى ضبط أداء هذا المؤقت .

المسجل TMR0 يمكن استخدامه كمؤقت يعد نبضات داخلية عبارة عن نبضة كل دورة أمر machine or instruction cycle والتى تساوى ربع نبضات التزامن Fosc/4 وفى حالة استخدامه كعداد فإنه يعد نبضات تزامن يتم إدخالها من خارج الشريحة على الطرف RA4/T0CKL . يتم ضبط أداء المؤقت أو العداد من خلال مجموعة من الأعلام الوجودة فى المسجل Option الذى سنقدمه فى الجزء التالى .

٤-٦-٦ المسجل OPTION-REG

هذا المسجل عنوانه هو 81h فى البانك 1 ، وهو يتكون من مجموعة من الأعلام التى تستخدم لضبط أداء المتحكم كما يلى وكما فى شكل (٤-٦) :

- **بت ٧** $\overline{RBPU}$: تستخدم لتنشيط أو إخماد مقاومات الجذب Pull up resistance على أطراف البوابة B . بوضع هذا الطرف بصفر يتم تنشيط هذه القاومات ، وبوضع هذا الطرف بواحد يتم إخماد هذه المقاومات .
- **بت ٦ INTEDG** : هذه البت خاصة بالمقاطعة الخارجية واختيار الحافة النشطة لها . بوضع هذه البت بواحد تنشط المقاطعة على الحافة الصاعدة للطرف RB0/INT ، بوضع هذه البت بصفر تنشط المقاطعة على الحافة النازلة .

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

شكل (٤-٦) المسجل option فى المتحكم PIC16f84

- **بت ٥ T0CS** : تستخدم لاختيار مصدر نبضات التزامن للمؤقت T0 . بوضعه يساوى واحد فإنه يتم اختيار المصدر الخارجى للنبضات من على الطرف RA4/T0CLK . بوضعه يساوى صفر يكون مصدر النبضات هو نبضات دورة الأمر Machine cycle الداخلية والتى تساوى ربع نبضات المذبذب Fosc/4 .

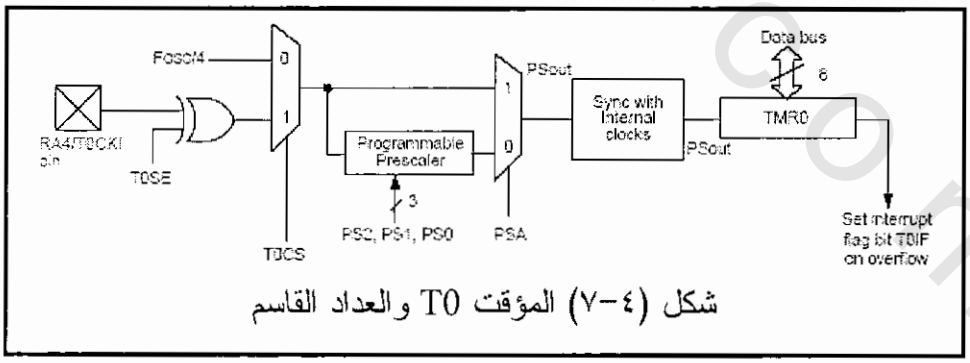
- **البت ٤ T0SE** ، عند إدخال نبضات تزامن خارجية من على الطرف ٥ ليعمل عليها المؤقت T0 ، فإنه يمكن اختيار الحافة الفعالة لهذه النبضات عن طريق هذا العلم . بوضع هذه البت تساوى واحد فإن المؤقت سيعمل عند الحافة النازلة (واحد إلى صفر) من نبضات التزامن الخارجية RA4/T0CLK ، وعند وضع هذه البت بصفر فإن المؤقت سيعمل عند الحافة الصاعدة (صفر إلى واحد) لهذه النبضات .
- **البت ٣ PSA** ، Prescaler assignment ، هناك عداد كما ذكرنا داخل المتحكم يستخدم كقاسم لنبضات التزامن حيث يتم قسمة نبضات التزامن على النسبة المضبوط عليها هذا العداد لإطالة زمن التوقيت . هذا العداد يمكن استخدامه مع المؤقت T0 أو مع مؤقت الحراسة WDT ولا يمكن استخدامه مع الإثنين فى وقت واحد . البت ٣ فى المسجل Option عند وضعها تساوى واحد فإن هذا العداد يستخدم كقاسم لنبضات تزامن مؤقت الحراسة WDT ، وعند وضعها بصفر فإن هذا العداد يستخدم كقاسم للمؤقت T0 .

Bit Value	Timer0 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

جدول (٤-١) ضبط نسبة القسمة للمؤقت T0 أو مؤقت الحراسة .

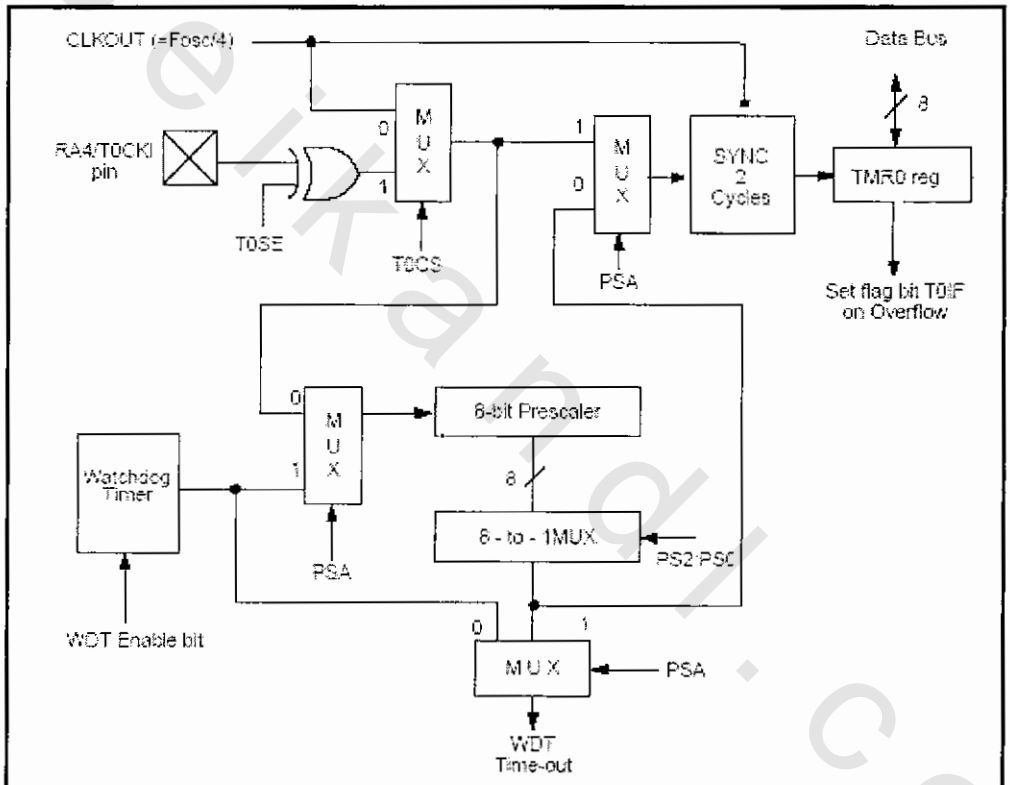
- **البتات ٢ ، ١ ، وصفر** ، هذه البتات تستخدم لضبط نسبة القسمة التى سيعل عندها العداد السابق . جدول (٤-١) يبين كيفية اختيار هذه النسبة سواء مع المؤقت T0 أو مع مؤقت الحراسة . لاحظ من هذا الجدول كيف يمكن إطالة زمن تأخير المؤقت T0 حتى نسبة ٢٥٦ مرة .

شكل (٤-٧) يبين عملية تخصيص العداد القاسم prescaler للمؤقت T0 أو لمؤقت الحراسة وتأثير الأعلام المختلفة فى المسجل Option على أداء المؤقت .



شكل (٤-٧) المؤقت T0 والعداد القاسم

لاحظ وجود فرصتى اختيار فى هذا الشكل ، الأولى نختار بها نبضات تزامن المؤقت T0 إما من الداخل (T0cs=0) أو من الخارج (T0cs=1) . الاختيار الثانى هو إما نضم العداد القاسم إلى T0 (PSA=0) ، أو ينضم إلى مؤقت الحراسة (PSA=1) . شكل (٤-٨) يبين عملية تخصيص العداد القاسم بتوضيح أكثر من شكل (٤-٧) .
عندما يكون العداد القاسم مخصصا للمؤقت T0 فإن أى عملية كتابة فى محتويات هذا المؤقت T0 ستسبب تصفيرا للعداد القاسم . يمكن الكتابة فى المؤقت T0 بأوامر مثل CLRF 1 ، الذى يصفر محتويات المؤقت T0 الذى عنوانه هو 01 ، والأمر MOVWF 1 الذى ينقل محتويات المسجل W إلى T0 ، والأمر BSF 1,x الذى يضع البت رقم x فى المسجل T0 بواحد .



شكل (٤-٨) رسم تفصيلى لتخصيص العداد القاسم للمؤقت T0 أو مؤقت الحراسة وعلاقة ذلك بالمسجل Option

عندما يكون العداد القاسم مخصصا لمؤقت الحراسة WDT فإن أى عملية كتابة فى محتويات هذا المؤقت WDT ستسبب تصفيرا للعداد القاسم . يمكن الكتابة فى المؤقت WDT بالأمر CLRWDT الذى يصفر محتويات مؤقت الحراسة وبالتالى يصفر العداد القاسم .

عملية تخصيص العداد القاسم للمؤقت T0 أو مؤقت الحراسة WDT تتم عن طريق البرمجة بعدد من الأوامر . مجموعة الأوامر التالية تنقل تبعية القاسم من المؤقت T0 إلى مؤقت الحراسة :

```
CLRWDT
CLRF  TMR0
MOVLW 'xxxx1xxx'b
OPTION
```

مجموعة الأوامر التالية تنقل تبعية القاسم إلى المؤقت T0 من مؤقت الحراسة :

```
CLRWDT
CLRF  TMR0
MOVLW 'xxxx0xxx'b
OPTION
```

عند فيضان مسجل المؤقت (المسجل TMR0) ، عندما يصل إلى FFh وعند نزوله إلى 00h فإن المتحكم يمكن أن يولد مقاطعة على حسب وضع بعض بتات مسجل المقاطعة الذى سنراه بعد قليل . البت T0IF فى المسجل INTCON<2> تمثل علم المقاطعة عند حدوث فيضان فى المؤقت T0 . هذا العلم يمكن حجبه أو السماح له عن طريق البت T0IE فى المسجل INTCON<5> . البت T0IF يجب أن تصفر عن طريق المستخدم فى برنامج خدمة المقاطعة قبل تنشيط أى مقاطعة أخرى من على المؤقت T0 .

٤-٦-٧ المسجل INTCON

هذا المسجل عنوانه هو 0Bh وهو موجود فى كل من البنك واحد وصفر فى ملف المسجلات . هذا المسجل يحتوى مجموعة من الأعلام التى تستخدم لتنشيط أو إخماد المقاطعات المختلفة . شكل (٤-٩) يبين محتويات هذا المسجل وسنبين فيما يلى وظيفة كل بت من بتاتها .

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
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

شكل (٤-٩) مسجل التحكم فى المقاطعة فى المتحكم PIC16f84

- **البت GIE** : منشط عام لكل أعلام المقاطعة . بوضع هذه البت بواحد يتم تنشيط جميع أعلام المقاطعة ، وبوضعه يساوى صفر يتم إخماد جميع أعلام المقاطعة .
- **البت EEIE** : علم مقاطعة الانتهاء من الكتابة فى ذاكرة البيانات EEPROM. بوضع هذه البت بواحد ينشط هذا العلم ليحدث مقاطعة عند الانتهاء من عملية الكتابة ، وبوضعه بصفر يتم إخماد هذا العلم .
- **البت T0IE** : علم تنشيط المقاطعة من المؤقت TMR0 ، بوضعه يساوى واحد تنشط المقاطعة من المؤقت ، بمعنى أنه عندما يحصل فيضان للمؤقت ، فإنه يقاطع المتحكم ، أما إذا وضعت هذه البت بصفر فإن هذه المقاطعة تكون مخمدة .
- **البت INTE** : علم تنشيط المقاطعة الخارجية على الطرف RB0/INT ، بوضع هذا الطرف بواحد فإن هذه المقاطعة تكون نشطة أو تقبل ، وبوضعه بصفر فإن المقاطعة على هذا الطرف تكون مخمدة .
- **البت RBIE** : تنشيط المقاطعة عند حدوث أى تغير فى جهد أى طرف من أطراف البوابة B . بوضع هذه البت بواحد فإنه عند حدوث أى تغير فى جهد البوابة B فإنه يحدث مقاطعة للمتحكم . بوضع هذه البت بصفر فإن هذه المقاطعة تكون مخمدة .
- **البت T0IF** : علم المقاطعة عند حدوث فيضان للمؤقت T0 . هذه البت تصبح واحد عند حدوث فيضان للمؤقت ، وتكون بصفر فى غير ذلك .
- **البت INTF** : هذه البت تصبح واحد عند حدوث مقاطعة على الطرف RB0/INT ، وتكون بصفر فى غير ذلك .
- **البت صفر RBIF** : علم المقاطعة عند حدوث تغير فى جهد أطرف البوابة B (RB4 إلى RB7) . هذه البت تصبح واحد عند حدوث تغير فى جهد هذه الأطراف ، وتكون بصفر فى غير ذلك .

لاحظ أن الأعلام على البتات صفر وواحد واثنين تكون بواحد عند حدوث المقاطعة القابلة لكل طرف ، ولكن الذى يقرر إذا كانت هذا المقاطعة تقبل أم لا هو حالة بت التنشيط لكل مقاطة مثل RBIE و INTE و T0IE .

٧-٤ مقاطعة المتحكم PIC16f84

يمكن مقاطعة المتحكم من أكثر من مصدر ، أولها المقاطعة من على الطرف RB0/INT حيث يمكن اختيار الحافة الفعالة لهذه المقاطعة باستخدام البت السادسة من المسجل option . عند حدوث حافة فعالة على الطرف RB0/INT فإن البت INTF ، INTCON<1> تصبح واحد . هذه المقاطعة يمكن حجبها عن طريق تصفير البت INTE ، INTCON<4> . البت INTF يجب تصفيرها عن طريق المستخدم فى برنامج خدمة المقاطعة حتى يمكن استقبال مقاطعة جديدة . المقاطعة على الطرف RB0/INT توقظ المتحكم من النوم لو أن البت INTE كانت بواحد .

المصدر الثانى لمقاطعة المتحكم هو حدوث فيضان للمؤقت TMR0 . عندما تصل بتات المؤقت إلى FFh ثم تنزل إلى 00h فإن العلم T0IF ، <INTCON<2 تصبح واحد . هذه المقاطعة يمكن حجبها أيضا عن طريق تصفير البت T0IE ، <INTCON<5 .

المصدر الثالث للمقاطعة هو تغير الجهد على الأطراف PB4 حتى PB7 . عند حدوث هذا التغير فإن البت RBIF ، <INTCON<0 تصبح واحد . يمكن حجب هذه المقاطعة أيضا بتصفير البت RBIE ، <INTCON<3 .

٨-٤ التعامل مع الذاكرة EEPROM

الذاكرة Electrically Erasable Programmable ROM, EEPROM هى ذاكرة بيانات كما ذكرنا سابقا يتم التعامل معها من خلال مسار بيانات ٨ بت حيث يمكن القراءة منها والكتابة فيها إلكترونيا . مقدار هذه الذاكرة فى المتحكم PIC16f84 هو ٦٤ بايت (من العنوان 00h حتى العنوان 3Fh) وهى ليست معنونة ضمن ملف المسجلات ولكن لها مدى عنوانى خاص بها . لذلك فإنها لا يمكن التعامل معها بالطريقة المباشرة ، ولكن يتم ذلك بطريقة غير مباشرة من خلال مجموعة من المسجلات الخاصة بهذا الغرض والموجودة ضمن ملف المسجلات . هذه المسجلات هى :

- المسجل EECON1 وعنوانه هو 88h فى البانك 1 .
- المسجل EECON2 وعنوانه هو 89h فى البانك 1 .
- المسجل EEDATA وعنوانه هو 08h فى البانك 0 .
- المسجل EEDADR وعنوانه هو 09h فى البانك 0 .

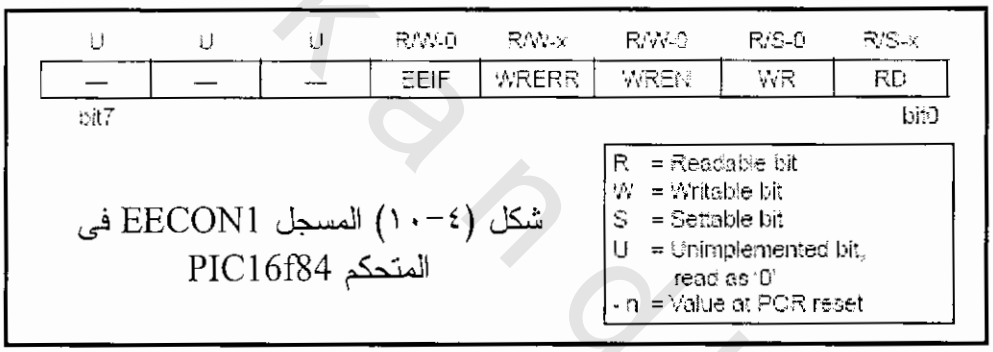
يتم وضع البيانات المراد كتابتها فى المسجل EEDATA ، كما أن البيانات التى سيتم قراءتها توضع أيضا فى هذا المسجل . عنوان البايت المراد التعامل معها يوضع فى المسجل EEADR . عملية الكتابة والقراءة يتم التحكم فيها من خلال المسجل EECON1 الذى سنعرض بتاته فيما يلى :

٨-٤-١ المسجل EECON1

وهو مسجل ٨ بت ، آخر ٣ بت فيه (البتات ٥ و ٦ و ٧) غير مستخدمه وعنوانه كما ذكرنا هو 88h فى البانك 1 وهو موضح فى شكل (٤-١٠) وباقى البتات وظائفها كا يلى :

- **البت ٤ EEIF** : علم مقاطعة عملية الكتابة فى الذاكرة EEPROM ، يضعه المتحكم بواحد للدلالة على الانتهاء من عملية كتابة ، ويظل صفرا طالما أن عملية الكتابة لم تنتهى . بعد الانتهاء من عملية الكتابة على المستخدم أن يصفر هذا العلم قبل الدخول فى دورة كتابة جديدة .

- **البت ٣ WRERR** : علم حدوث خطأ فى عملية الكتابة ، يصبح واحد عندما تنتهى عملية الكتابة بطريقة غير طبيعية بسبب حدوث خطأ ما . يكون بصفر عند انتهاء عملية الكتابة بطريقة طبيعية .
- **البت ٢ WREN** : تنشيط عملية الكتابة حيث يجب عن طريق المستخدم وضع هذه البت بواحد حتى تنشط أو يسمح بعملية الكتابة . هذه البت هى بمثابة حجاب لعملية الكتابة . إذا وضعت هذه البت بصفر فإن عملية الكتابة تحجب أو لا يسمح لها .
- **البت ١ WR** : بت التحكم فى الكتابة ، توضع بواحد عن طريق المبرمج للبدأ فى عملية الكتابة ، عند الانتهاء من الكتابة فإنها ترجع للصفر مرة أخرى دلالة على الانتهاء من الكتابة .
- **البت صفر RD** : بت التحكم فى القراءة ، توضع بواحد عن طريق المبرمج للبدأ فى عملية القراءة ، عند الانتهاء من القراءة فإنها ترجع للصفر مرة أخرى دلالة على الانتهاء من عملية القراءة .



شكل (٤-١٠) المسجل EECON1 فى المتحكم PIC16f84

مثال على عملية القراءة من الذاكرة EEPROM

لكى تتم عملية القراءة من هذه الذاكرة فإنه على المبرمج أن يضع العنوان المراد قراءته فى المسجل EEADR ، وبعدها يجعل بت التحكم فى القراءة RD فى المسجل EECON1<0> بواحد ، حيث بعدها سيجد البيانات المراد قراءتها فى المسجل EEDATA . البرنامج التالى يوضح هذه العلية :

```
BCF   STATUS, RP0 ;  إختيار البانك صفر
MOVLW      ADDRESS ;  العنوان المراد التعامل معه فى مسجل التشغيل
MOVWF      EEADR
BSF        STATUS,RP0 ;  إختيار البانك واحد
BSF        EECON1,RD  ;  بدأ عملية القراءة
BCF        STATUS,RP0
MOVF EEDATA,W   ;  وضع البيانات فى مسجل التشغيل
```

مثال على عملية الكتابة فى الذاكرة EEPROM

للكتابة فى أى عنوان فى الذاكرة EEPROM فإن المستخدم يبدأ بوضع العنوان الذى سيتم التعامل معه فى المسجل EEADR ، ثم يضع البيانات فى المسجل EEDATA ثم يتبع الخطوات التالية فى كل عملية كتابة :

```
BSF     STATUS,RP0
BCF     INTCON,GIE
BSF     ECON1,WREN
MOVLW   DATA
MOVWF   EEDATA
MOVLW   55H
MOVWF   EECON2
MOVLW   AAH
MOVWF   EECON2
BSF     EECON1,WR
BSF     INTCON,GIE
```

عملية الكتابة لن تتم إلا إذا تم اتباع التتابع السابق من الأوامر لكل عملية كتابة فى هذا النوع من الذاكرة . نلاحظ من هذا التتابع أن جميع المقاطعات يتم حجبها أثناء عملية الكتابة عن طريق وضع البت GIE فى المسجل INTCON بصفر فى بداية عملية الكتابة .

٤-٩ مجموعة أوامر المتحكم PIC16f84

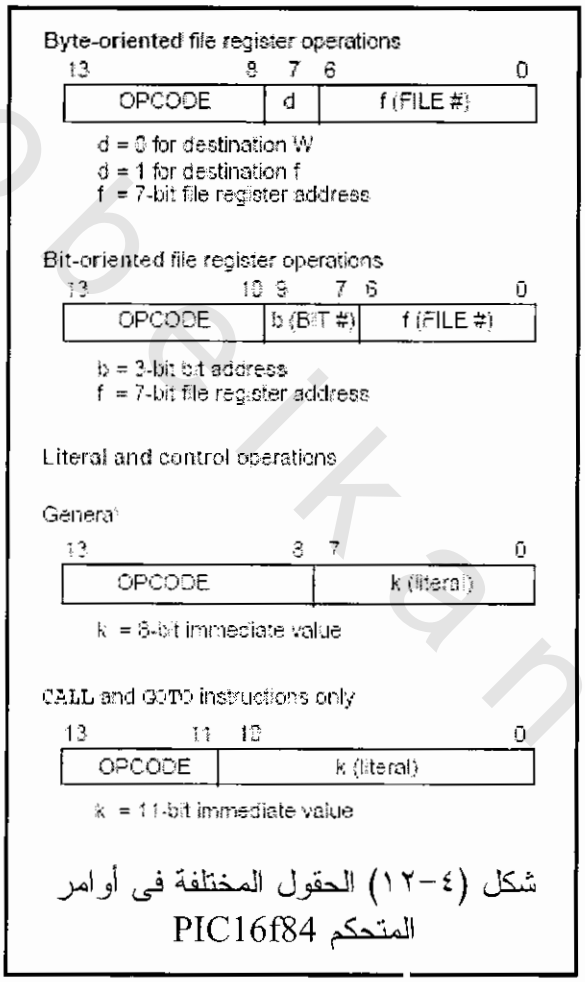
شكل (٤-١١) يبين بعض الاختصارات التى سيتم استخدامها فى شرح مجموعة الأوامر الخاصة بهذا المتحكم .

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1

شكل (٤-١١) الاختصارات المستخدمة فى الأوامر

كل واحد من أوامر المتحكمات PIC يتكون من بايت واحدة وهذه البايت تكون ١٢ أو ١٤ أو ١٦ بت على حسب نوع المتحكم . المتحكم PIC16f84 تتكون كل أوامره من ١٤ بت . شكل (٤-١٢) يبين الشكل العام لهذه الأوامر حيث نلاحظ أنها تتكون من أكثر من حقل . الحقل الأول وهو مجموعة البتات ذات القيمة

الأعلى (فى أقصى يسار الأمر) تمثل شفرة الأمر Opcode التى تبين للمتحكم نوع الأمر والكثير من المعلومات الأخرى عن هذا الأمر . الحقل الثانى يمثل عنوان المسجل الذى سيتعامل معه هذا الأمر وذلك فى الأوامر التى تتعامل على مستوى البايت أو تتعامل مع مسجلات . هذا الحقل يتكون من ٧ بت مما يعنى أن هناك ١٢٨ مسجلا يمكن التعامل معها كما رأينا . فى الأوامر التى تتعامل على مستوى البايت هناك حقل يتكون من بت واحدة d تحدد الهدف أو المكان الذى ستذهب إليه النتيجة . إذا كانت البت d=0 فإن ذلك يعنى ان النتيجة ستذهب إلى مسجل التشغيل W . أما إذا كانت d=1 فإن ذلك يعنى أن النتيجة ستذهب إلى المسجل الآخر الموجود فى الأمر . فى حالة الأوامر التى تتعامل على مستوى البت فإن البت التى سيتم التعامل معها يوضع الكود الخاص بها فى حقل مكون من ٣ بت كما فى شكل (٤-١٢) . الأوامر التى تتعامل مع ثوابت Literals أو أوامر القفز يتم وضع الثابت أو العنوان الذى سيتم القفز إليه فى حقل خاص بذلك كما فى الشكل .



شكل (٤-١٢) الحقول المختلفة فى أوامر المتحكم PIC16f84

جدول (٤-٢) يبين جميع أوامر المتحكم PIC16f84 وهى نفسها أوامر المتحكمات PIC12cXX . لاحظ أنها مقسمة إلى ثلاث مجموعات ، المجموعة الأولى هى مجموعة الأوامر التى تتعامل على مستوى البايت Byte oriented ، والمجموعة الثانية هى التى تتعامل على مستوى البت Bit oriented ، والمجموعة الثالثة هى التى تتعامل مع الثوابت وأوامر القفز .

العمود الأول يبين شفرة الأسمبلى لكل أمر ، والعمود الثانى يبين شرحا مختصرا فى عبارة واحدة لما يفعله كل أمر . العمود الثالث يبين الحقول المختلفة وعدد البتات فى كل حقل من حقول كل أمر . العمود الرابع يبين الأعلام التى تتأثر عند تنفيذ كل واحد من هذه الأوامر .

جدول (٤-٣) يبين مجموعة أوامر المتحمات PIC البسيطة والمتوسطة الإمكانيات مرتبة ترتيبا أبجديا مع شرح مختصر ومثال لكل أمر .

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

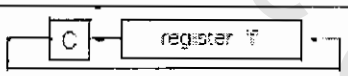
جدول (٤-٢) مجموعة أوامر المتحكمات PIC البسيطة والمتوسطة الإمكانيات

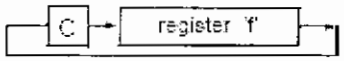
جدول (٤-٣) أوامر المتحكمات PIC البسيطة والمتوسطة مرتبة ترتيبا أبجديا

الأمر	العملية	وصف الأمر	مثال
ADDWF	(W)+(f) → (dest)	يجمع محتويات المسجل W مع المسجل f والنتيجة تذهب للهدف على حسب قيمة d .	**ADDWF FSR,0** **Before:** W = 0x17 FSR = 0xC2 **After:** W = 0xD9 FSR = 0xC2
ANDLW	(W).AND.(k) → (W)	عملية AND لمحتويات المسجل W مع ثابت والنتيجة في المسجل W.	**Before:** W = 0xA3 **After:** W = 0x03
ANDWF	(W).AND.(f)→ (dest)	عملية AND لمحتويات المسجل W مع المسجل f والنتيجة تذهب للهدف على حسب قيمة d .	**Before:** W = 0x17 FSR = 0xC2 **After:** W = 0x17 FSR = 0x02
BCF	0 → (f<b>)	البت b في المسجل f يتم تصفيرها .	**Before:** FLAG_REG = 0xC7 **After:** FLAG_REG = 0x47
BSF	1 → (f<b>)	البت b في المسجل f يتم وضعها بواحد .	**Before:** FLAG_REG = 0x0A **After:** FLAG_REG = 0x8A
BTFSC	skip if (f<b>) = 0	إذا كانت البت b في المسجل f تساوى صفر فإن الأمر التالى يترك أو لا ينفذ .	
BTFSS	skip if (f<b>) = 1	إذا كانت البت b في المسجل f تساوى واحد فإن	

الأمر	العملية	وصف الأمر	مثال
		الأمر التالى يترك أو لا ينفذ .	
CALL	(PC)+1→ Top of Stack; k → PC<7:0>; (STATUS<6:5>) → PC<10:9>; 0 → PC<8>	نداء على برنـــامج فرعى	
CLRF	00h → (f); 1 → Z	تصفير محتويـــات المسجل f ووضع واحـــد فـــى علـــم الصفر .	**Before:** FLAG_REG = 0x5A **After:** FLAG_REG = 0x00 Z = 1
CLRW	00h → (W); 1 → Z	تصفير محتويـــات المســـــــجل W ووضع واحد فـــى علم الصفر .	**Before:** W = 0x5A **After:** W = 0x00 Z = 1
CLRWDT	00h → WDT; 0 → WDT prescaler (if assigned); 1 → TO; 1 → PD	تصفير محتويـــات مؤقت الحراسة .	**Before:** WDT counter = ? **After:** WDT counter = 0x00 WDT prescale = 0 TO = 1 PD = 1
COMF	(/f) → (dest)	عكس محتويـــات المسجل f تـــذهب إلى الهدف علـــى حسب قيمة d .	**Before:** REG1 = 0x13 **After:** REG1 = 0x13 W = 0xEC
DECF	(f) – 1 → (dest)	إنقاص واحد مـــن المسجل f والنتيجة تذهب للهدف على حسب قيمة d .	**Before**: CNT = 0x01 Z = 0 **After:** CNT = 0x00 Z = 1

مثال	وصف الأمر	العملية	الأمر
	إنقاص واحد من المسجل f والنتيجة تذهب للهدف على حسب قيمة d ، والأمر التالى لن ينفذ إذا كانت النتيجة بصفر .	(f) – 1 → d; skip if result = 0	**DECFSZ**
After: PC = address	قفز غير مشروط للعنوان K .	k → PC<8:0>; STATUS<6:5> → PC<10:9>	**GOTO**
Before: CNT = 0xFF Z = 0 **After:** CNT = 0x00 Z = 1	زيادة واحد على محتويات المسجل f والنتيجة تذهب للهدف على حسب قيمة d .	(f) + 1 → (dest)	**INCF**
	زيادة واحد على محتويات المسجل f والنتيجة تذهب للهدف على حسب قيمة d ، والأمر التالى لن ينفذ إذا كانت النتيجة بصفر .	(f) + 1 → (dest), skip if result = 0	**INCFSZ**
Before: W = 0x9A **After:** W = 0xBF Z = 0	عملية OR لمحتويات المسجل W مع ثابت والنتيجة فى المسجل W.	(W).OR. (k) → (W)	**IORLW**
Before: RESULT = 0x13 W = 0x91 **After:** RESULT = 0x13	عملية OR لمحتويات المسجل W مع المسجل f والنتيجة تذهب للهدف على حسب قيمة d .	(W).OR. (f) → (dest)	**IORWF**

مثال	وصف الأمر	العملية	الأمر
W = 0x93 Z = 0			
After: W = value in FSR register	محتويات المسجل f تـذهب للهـدف على حسب قيمة d .	(f) → (dest)	**MOVF**
After: W = 0x5A	الثابـت k يـذهب للمسجل w .	k → (W)	**MOVLW**
Before: TEMP_REG = 0xFF W = 0x4F **After:** TEMP_REG = 0x4F W = 0x4F	محتويات المسجل W تذهب للمسجل f .	(W) → (f)	**MOVWF**
No operation	لا تعمل شيء	No operation	**NOP**
Before: W = 0x07 **After:** OPTION = 0x07	محتويات المسجل W تـذهب إلـى المسجل option	(W) → OPTION	**OPTION**
	عودة من برنـامج فرعى ، مع تحميل السجل w بالثابت k .	k → (W); TOS → PC	**RETLW**
Before: REG1 = C = **After:** REG1 = W = C =	دوران محتويـات المسجل f ناحيـة اليسار بت واحدة ، والنتيجـة تـذهب للهدف على حسب قيمة d .		**RLF**

الأمر	العملية	وصف الأمر	مثال
RRF		دوران محتويات المسجل f ناحية اليمين بت واحدة ، والنتيجة تذهب للهدف على حسب قيمة d .	**Before:** REG1 = C = **After:** REG1 = W = C =
SLEEP	00h → WDT; 0 → WDT prescaler; 1 → TO; 0 → PD	يدخل المتحكم فى حالة نوم .	
SUBWF	(f) – (W) → (dest)	محتويات المسجل w يتم طرحها من المسجل f والنتيجة تذهب للهدف المحدد بقيمة d .	**Before**: REG1 = 3 W = 2 C = ? **After**: REG1 = 1 W = 2 C = 1 ; result is positive **Example 2:** **Before**: REG1 = 2 W = 2 C = ? **After**: REG1 = 0 W = 2 C = 1 ; result is zero **Example 3:** **Before:** REG1 = 1 W = 2 C = ? **After**: REG1 = FF W = 2 C = 0 ; result is negative

الأمر	العملية	وصف الأمر	مثال
SWAPF	(f<3:0>) → (dest<7:4>); (f<7:4>) → (dest<3:0>)	النصف الأول من المسجل f يستبدل مع النصف الثانى من نفس المسجل ، النتيجة تذهب للهدف المحدد بقيمة d .	**Before:** REG1 = 0xA5 **After:** REG1 = 0xA5 W = 0X5A
TRIS	(W) → TRIS register f	محتويات المسجل w تذهب إلى مسجل الفصل TRIS . هناك TRISA و TRISB فى المتحكم 16f84A.	**Before:** W = 0XA5 **After:** TRIS = 0XA5 **Note:** f = 6 for PIC12C5XX only.
XORLW	(W) .XOR. k → (W)	عملية XOR لمحتويات المسجل W مع ثابت والنتيجة فى المسجل W.	**Before:** W = 0xB5 **After:** W = 0x1A
XORWF	(W) .XOR. (f) → (dest)	عملية XOR لمحتويات المسجل W مع المسجل f والنتيجة تذهب للهدف على حسب قيمة d .	**Before:** REG = 0xAF W = 0xB5 **After:** REG = 0x1A W = 0xB5

الفصل القادم سيحتوى أمثلة على برمجة هذا النوع من المتحكمات .